Lexique

Bialasik Théo

Sommaire

HTML

CSS

JS

REACT

Node

BDD

HTML

Introduction au HTML

HTML : HyperText Markup Language (Langage de balisage hypertexte)

W3C : World Wide Web Consortium (Organisme de standardisation du web)

WHATWG : Web Hypertext Application Technology Working Group (Groupe de travail sur les technologies web)

DOM : Document Object Model (Modèle objet du document)

Structure de base du HTML

DOCTYPE : Déclaration du type de document

<html> : Élément racine d'un document HTML

<head> : Contient les métadonnées du document

<body> : Contient le contenu visible de la page

<meta> : Métadonnées de la page

<title> : Titre de la page

Les balises de contenu

- <h1> à <h6> : Titres de différents niveaux
- <p> : Paragraphe
-
 : Saut de ligne
- <hr> : Ligne horizontale
- : Conteneur de division
- : Élément en ligne

Les éléments de liste

- : Liste non ordonnée
- : Liste ordonnée
- : Élément de liste

Les liens et navigations

<a> : Lien hypertexte (anchor)

URL : Uniform Resource Locator (Adresse d’une ressource sur le web)

HREF : Hypertext REFerence (Attribut de lien)

<nav> : Section de navigation

Les images et médias

- <img> : Image
  - SRC : Source (Chemin de l'image)
  - ALT : Alternative Text (Texte alternatif)
- <video> : Vidéo
- <audio> : Audio
- <figure> : Figure illustrative
- <figcaption> : Légende d'une figure

Les formulaires et entrées utilisateur

- <form> : Formulaire
- <input> : Champ de saisie
  - TYPE : Détermine le type de champ (text, email, password, etc.)
- <label> : Étiquette associée à un champ
- <button> : Bouton
- <select> : Liste déroulante
- <textarea> : Zone de texte multi-lignes
- POST : Méthode HTTP pour envoyer des données
- GET : Méthode HTTP pour récupérer des données

Les tableaux

- <table> : Table
- <tr> : Ligne de tableau
- <td> : Cellule de tableau
- <th> : Cellule d’en-tête
- <thead> / <tbody> / <tfoot> : Sections de tableau

Les éléments sémantiques

- <header> : En-tête de page
- <footer> : Pied de page
- <article> : Article autonome
- <section> : Section thématique
- <aside> : Contenu complémentaire
- <main> : Contenu principal

Les éléments interactifs

- <details> : Détails interactifs
- <summary> : Résumé d'un bloc de détails
- <dialog> : Boîte de dialogue modale

Les attributs globaux importants

ID : Identifiant unique

CLASS : Classe pour le style CSS

STYLE : Style en ligne

LANG : Langue du document

DATA-* : Attributs personnalisés de données

Concepts avancés

HTML5 : Dernière version majeure de HTML

SEO : Search Engine Optimization (Optimisation pour les moteurs de recherche)

Responsive : Conception adaptative

ARIA : Accessible Rich Internet Applications (Accessibilité)

MIME : Multipurpose Internet Mail Extensions (Type de contenu)

CSS

Introduction au CSS

- CSS : Cascading Style Sheets (Feuilles de style en cascade)
- W3C : World Wide Web Consortium (Organisme de standardisation)
- Selectors : Sélecteurs pour cibler les éléments HTML
- Properties : Propriétés pour définir le style des éléments
- Values : Valeurs associées aux propriétés CSS
- Syntax : Structure d'une règle CSS

Syntaxe de base du CSS

- Sélecteur : Cible un élément (h1, .class, #id)
- Propriété : Définit le style (color, font-size)
- Valeur : Définit les paramètres d'une propriété (red, 16px)
- Déclaration : Combinaison d’une propriété et d’une valeur (color: blue;)
- Règle CSS : Ensemble de sélecteur(s) et de déclaration(s)

Types de sélecteurs CSS

Sélecteurs simples :

- * : Sélecteur universel
- element : Sélecteur de type (p, div)
- .class : Sélecteur de classe
- #id : Sélecteur d'ID
- [attribute] : Sélecteur d'attribut

Sélecteurs combinés :

- A B : Sélecteur descendant (A contient B)
- A > B : Sélecteur enfant direct
- A + B : Sélecteur de frère adjacent
- A ~ B : Sélecteur de frère général

Pseudo-classes :

- :hover : Lorsqu'un élément est survolé
- :focus : Lorsqu'un élément est sélectionné
- :nth-child(n) : Cible un élément parmi plusieurs

Pseudo-éléments :

- ::before : Insère un contenu avant un élément
- ::after : Insère un contenu après un élément

Modèle de boîte CSS (Box Model)

- Content : Contenu de l'élément
- Padding : Espacement interne
- Border : Bordure autour de l'élément
- Margin : Espacement externe
- Width/Height : Largeur et hauteur de l'élément
- Valeurs : px, em, %, vw, vh, etc.

Propriétés CSS courantes

- Texte : color, font-size, font-family, text-align, line-height
- Arrière-plan : background-color, background-image, background-size
- Bordures : border, border-radius
- Espacement : margin, padding
- Positionnement : position, top, left, z-index
- Affichage : display, visibility, opacity
- Flexbox : display: flex, justify-content, align-items
- Grid : display: grid, grid-template-columns, grid-gap

Unités de mesure en CSS

Unités absolues :

- px (pixels),
- cm,
- mm,
- in

Unités relatives :

- em : Par rapport à la taille de la police de l'élément parent
- rem : Par rapport à la racine du document
- % : Pourcentage de l'élément parent
- vw / vh : Pourcentage de la largeur/hauteur de la fenêtre

Positionnement et affichage

Position :

- static (par défaut)
- relative (relative à sa position normale)
- absolute (par rapport à son ancêtre positionné)
- fixed (positionné par rapport à la fenêtre)
- sticky (fixé en fonction du défilement)

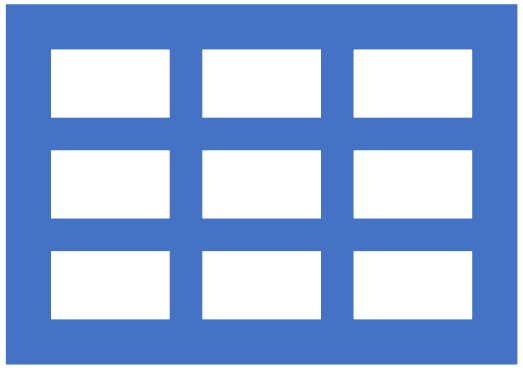
Display :

- block (occupe toute la largeur)
- inline (occupe l'espace du contenu)
- inline-block (mix des deux)
- flex (disposition flexible)
- grid (grille de mise en page)
- none (masquer l'élément)

Flexbox (Disposition flexible)

- Propriétés principales :
  - display: flex;
  - flex-direction : row, column
  - justify-content : flex-start, center, space-between
  - align-items : flex-start, center, stretch
  - flex-wrap : nowrap, wrap

Grid Layout (Mise en page en grille)



- Propriétés de base :
  - display: grid;
  - grid-template-columns / grid-template-rows
  - grid-gap : Espacement entre les cellules
  - grid-area : Attribution d'une zone spécifique

Effets et animations

Transitions :

- transition: all 0.3s ease-in-out;

Animations :

- @keyframes : Définition des étapes d'animation
- animation : Associer l'animation à un élément

Transformations :

- transform: rotate(45deg);
- scale(), translate(), skew()

Média queries et responsive design

Média queries : Permettent l'adaptation aux différentes tailles d'écran

Breakpoints courants :

Mobile : max-width: 576px
Tablette : max-width: 768px
Ordinateur : min-width: 1024px

Préprocesseurs CSS

- SASS (Syntactically Awesome Stylesheets)
- LESS (Leaner CSS)
- SCSS (Syntaxe avancée de SASS)
- Variables, mixins, et nesting pour écrire du CSS plus efficace

Bonnes pratiques CSS

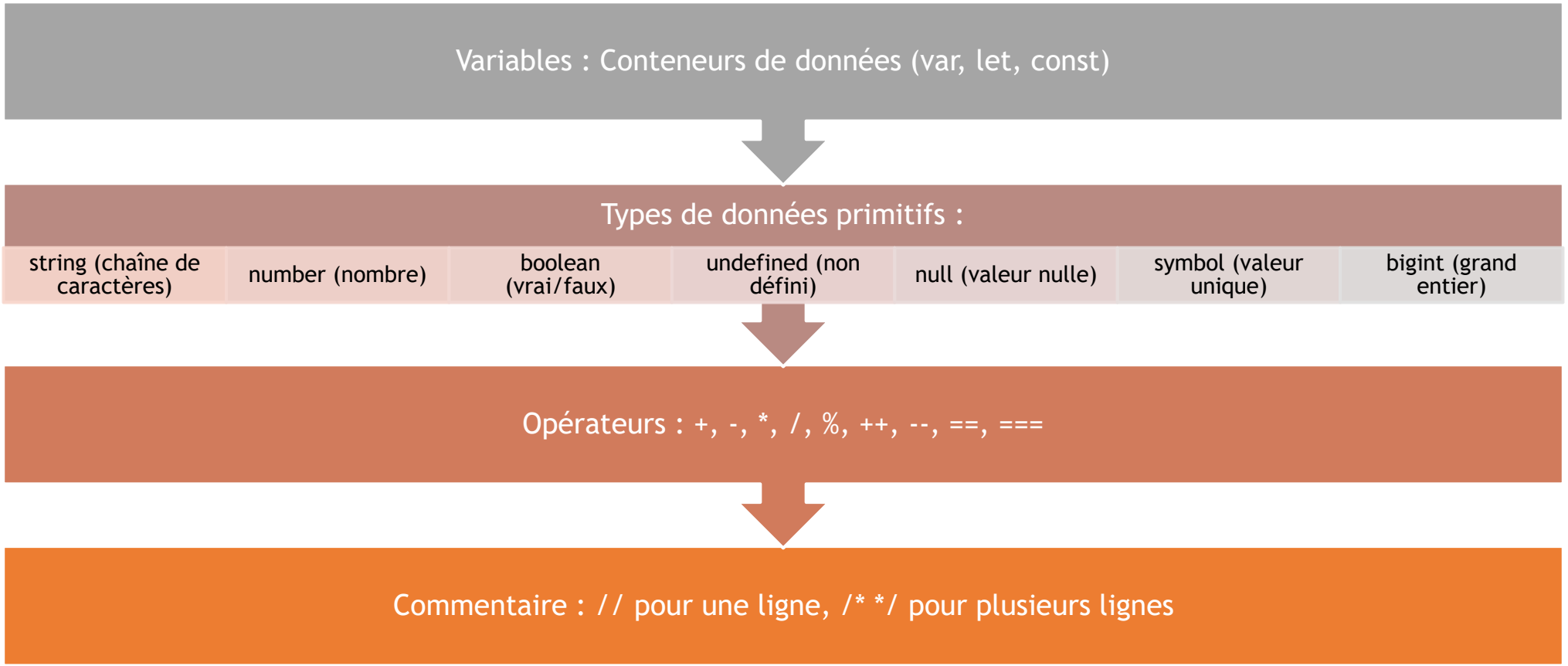
- DRY (Don't Repeat Yourself) : Éviter la redondance
- BEM (Block Element Modifier) : Convention de nommage
- Minification : Réduction de la taille des fichiers CSS
- Utilisation de variables avec :root

- JS

Introduction à JavaScript

- JS : Abréviation courante de JavaScript
- ECMAScript (ES) : Norme sur laquelle JavaScript est basé
- ES6+ : Versions modernes de JavaScript à partir d'ECMAScript 2015
- TC39 : Comité responsable de l'évolution du langage
- V8 : Moteur JavaScript utilisé par Google Chrome et Node.js
- Node.js : Environnement d'exécution côté serveur pour JS

Syntaxe de base



Structures de contrôle

Conditions :

if, else if, else

Opérateur ternaire : condition ? valeur1 : valeur2

switch (choix multiples)

Boucles :

for (itération contrôlée)

while (tant qu'une condition est vraie)

do...while (exécute au moins une fois)

forEach() (pour parcourir un tableau)

Fonctions

- Déclaration : function nom() {}
- Expression : const nom = function() {}
- Fléchée (ES6) : const nom = () => {}
- Paramètres par défaut : function(x = 10) {}
- Rest parameter : function(...args) {}
- Closures : Fonction imbriquée qui se souvient de son contexte parent
- Callback : Fonction passée en argument à une autre fonction

Portée et hoisting

Scope (portée) :

Global : Accessible partout

Local (function scope) : Définie à l'intérieur d'une fonction

Block scope (ES6) : Avec let et const

Hoisting : Mécanisme de déplacement des déclarations en haut du script

Objets et tableaux

Objet : Collection de paires clé-valeur

Tableau : Liste ordonnée d'éléments

Méthodes courantes :

- Objets : Object.keys(), Object.values(), Object.assign()
- Tableaux : push(), pop(), map(), filter(), reduce()

Programmation orientée objet (POO)

- **Classes (ES6)** : Modèle de création d'objets
- Prototype : Mécanisme d'héritage en JavaScript
- Héritage : extends pour hériter d'une autre classe
- Encapsulation : Utilisation de propriétés privées avec #

Asynchronisme et gestion des promesses

Asynchrone vs Synchrone :

Synchrone : Exécution ligne par ligne

Asynchrone : Exécution non bloquante (ex. requêtes réseau)

Callbacks : Fonction passée pour gérer l'asynchronisme

Promises : Objet représentant la réussite ou l'échec d'une opération

Async/Await : Syntaxe plus lisible pour gérer les promesses

Manipulation du DOM (Document Object Model)

Sélection d'éléments :

- document.getElementById('id')
- document.querySelector('.class')
- document.querySelectorAll('div')

Modification :

- element.textContent (changer le texte)
- element.innerHTML (changer le HTML)
- element.style.color = "red" (changer le style)

Événements :

- addEventListener('click', function)
- removeEventListener()

Gestion des événements

Types d'événements : click, mouseover, keydown, submit

Propagation :

bubbling (de l'enfant au parent)
capturing (du parent à l'enfant)

Prévention du comportement par défaut : event.preventDefault()

Gestion des erreurs

TRY-CATCH : GESTION DES ERREURS

FINALLY : BLOC EXÉCUTÉ APRÈS TRY/CATCH, QU'IL Y AIT UNE ERREUR OU NON

THROW : LANCER UNE EXCEPTION MANUELLEMENT

API Web et AJAX

Fetch API : Récupération de données via HTTP

XHR (XMLHttpRequest) : Ancienne méthode d'appel AJAX

LocalStorage / SessionStorage : Stockage persistant côté client

Cookies : Petits fichiers de stockage de données sur le navigateur

Frameworks et bibliothèques populaires

React.js : Bibliothèque pour interfaces utilisateur

Vue.js : Framework progressif

Angular : Framework complet de Google

jQuery : Bibliothèque pour simplifier le DOM (anciennement populaire)

Bonnes pratiques et outils

- Linting : ESLint pour détecter les erreurs de code
- Minification : Réduction de la taille du code pour la performance
- Débogage : console.log(), debugger
- Test unitaire : Jest, Mocha, Jasmine

REACT

Introduction à React

- React.js : Bibliothèque JavaScript pour la création d'interfaces utilisateur
- JSX (JavaScript XML) : Syntaxe permettant d'écrire du HTML dans du JavaScript
- ReactDOM : API pour manipuler le DOM avec React
- SPA (Single Page Application) : Application web où la navigation ne nécessite pas de rechargement de page
- Virtual DOM : Représentation en mémoire du DOM réel pour des mises à jour efficaces

Principes de base de React

Composants : Éléments réutilisables définissant l'interface utilisateur

- Fonctionnels (function Component() {})
- Classiques (class Component extends React.Component)

Props (Propriétés) : Données passées aux composants pour la personnalisation

State (État) : Données internes d’un composant qui peuvent changer au fil du temps

Rendu conditionnel : Affichage dynamique basé sur certaines conditions

JSX (JavaScript XML)

Extension de syntaxe qui permet d'écrire du HTML dans le code JavaScript

Doit être transformé en code JS pur par Babel

Les Composants React

Fonctionnels :

- function Welcome(props) { return <h1>Hello, {props.name}</h1>; }

Classiques (anciennement utilisés) :

- class Welcome extends React.Component { render() { return <h1>Hello, {this.props.name}</h1>; } }

Props (Propriétés) : Données passées d'un composant parent à enfant

Children : Contenu passé entre les balises d'un composant

Gestion de l'état (State Management)

useState (Hook) : Gérer l'état dans un composant fonctionnel

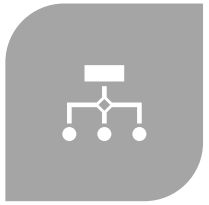
setState (Class Components) : Mise à jour de l’état dans les composants de classe

Lifting State Up : Faire remonter l’état au composant parent pour un contrôle centralisé

Props Drilling : Transmission excessive des props à travers plusieurs niveaux

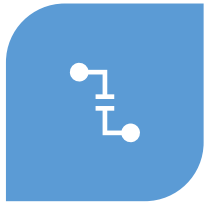
Hooks React *(Introduits dans React 16.8)*

USESTATE : GESTION DE L'ÉTAT LOCAL



USEEFFECT : EFFETS SECONDAIRES DANS UN COMPOSANT (COMME COMPONENTDIDMOUNT)

USECONTEXT : GESTION DU CONTEXTE GLOBAL SANS PROPS DRILLING



USEREF : RÉFÉRENCES POUR MANIPULER LE DOM OU STOCKER DES VALEURS PERSISTANTES

USEREDUCER : GESTION D'ÉTAT AVANCÉE (ALTERNATIVE À REDUX)

USEMEMO : MÉMORISATION DES CALCULS INTENSIFS

USECALLBACK : MÉMORISATION DES FONCTIONS POUR ÉVITER LES RE-RENDER INUTILES

Cycle de vie des composants (Lifecycle Methods) (Class Components uniquement)

Montage (Mounting) : componentDidMount()

Mise à jour (Updating) : componentDidUpdate()

Démontage (Unmounting) : componentWillUnmount()

Erreur (Error Handling) : componentDidCatch()

Gestion des événements

Gestion des événements similaire au DOM standard avec la syntaxe camelCase

Accès à l'objet événement via event dans la fonction de gestion

Listes et rendu dynamique

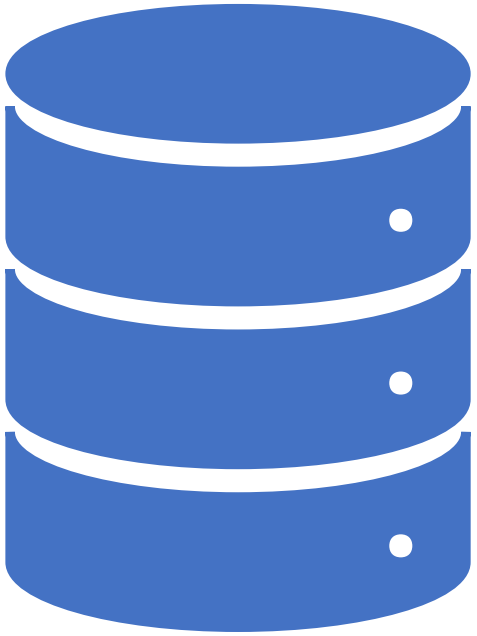
UTILISATION DE MAP() POUR GÉNÉRER DES LISTES D'ÉLÉMENTS

CLÉ UNIQUE (KEY) : AIDE REACT À IDENTIFIER CHAQUE ÉLÉMENT DE LA LISTE

Formulaires en React

Gestion des champs avec useState

Formulaires contrôlés vs non contrôlés



Context API (Gestion d'état globale sans Redux)

Fournit un moyen de gérer l'état à travers l'application

React Router (Gestion de la navigation)

Bibliothèque pour gérer la navigation côté client

Composants principaux :

<BrowserRouter> : Conteneur de la navigation

<Route> : Définit une route

<Link> : Crée un lien de navigation

Gestion d'état avancée

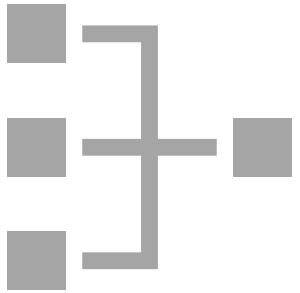
Redux : Bibliothèque populaire pour la gestion d’état global

Zustand : Gestion d'état plus légère

Recoil : Solution développée par Facebook pour React

Effets et animations

Framer Motion : Librairie pour les animations avancées



React Transition Group : Gestion des animations lors des montages/démontages de composants

Tests en React

Jest : Framework de test

React Testing Library : Tests orientés utilisateur

Exécution de tests d'interface utilisateur avec render() et screen.getByText()

Optimisation des performances

- Memoization avec React.memo pour éviter les re-renders inutiles
- Lazy Loading avec React.lazy pour charger des composants à la demande
- Suspense pour afficher un fallback en attente du chargement

Déploiement et build

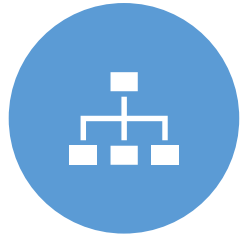
- Utilisation de Create React App (CRA) pour générer une application prête à l'emploi
- Vite.js pour un environnement de développement rapide
- Déploiement sur Netlify, Vercel, GitHub Pages, etc.
- Commandes essentielles
  - Npm run build
  - Npm start

Bonnes pratiques React

COMPOSANTS RÉUTILISABLES ET DÉCOUPLAGE DU CODE

UTILISATION DE TYPESCRIPT POUR LA SÉCURITÉ DES TYPES

APPLICATION DES DESIGN PATTERNS COMME HOC (HIGHER ORDER COMPONENTS) ET RENDER PROPS



ORGANISATION DES FICHIERS SELON UNE ARCHITECTURE MODULAIRE

- Node

Introduction à Node.js

- Node.js : Environnement d'exécution JavaScript côté serveur basé sur le moteur V8 de Google Chrome
- V8 : Moteur JavaScript de Google utilisé dans Node.js
- Event-Driven : Architecture basée sur la gestion des événements asynchrones
- Non-Blocking I/O : Traitement asynchrone des opérations d'entrée/sortie
- Single-Threaded : Modèle d'exécution en un seul thread avec une boucle d'événements

Installation et gestion des packages

NPM (Node Package Manager) : Gestionnaire de packages officiel pour Node.js

- Installation de packages : npm install package-name
- Fichier de dépendances : package.json

Yarn : Alternative à NPM pour la gestion des dépendances

nvm (Node Version Manager) : Outil pour gérer plusieurs versions de Node.js

CommonJS : Système de modules par défaut (require)

ESM (ECMAScript Modules) : Modules modernes (import/export)

Modules en Node.js

Importation avec CommonJS

- const fs = require('fs’);

Exportation avec CommonJS

- module.exports = { myFunction };

Importation avec ESM

- import fs from 'fs’;

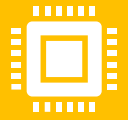
Modules intégrés (Core Modules) :

- fs (File System)
- http (Serveur HTTP)
- path (Manipulation des chemins de fichiers)
- os (Informations système)
- util (Outils divers)

Gestion des fichiers et du système (fs, path, os)

fs (File System) : Manipulation des fichiers

path : Manipulation des chemins de fichiers



os : Informations système (CPU, mémoire, etc.)

Gestion des requêtes HTTP

GET, POST, PUT, DELETE : Méthodes HTTP courantes

req (Request) : Requête entrante contenant les données

res (Response) : Réponse envoyée au client

Middleware : Fonction exécutée entre la requête et la réponse

Middleware dans Express.js

- Middleware intégré : express.json() pour le parsing JSON
  - app.use((req, res, next) => { console.log('Middleware exécuté'); next(); })

Base de données avec Node.js

MySQL/PostgreSQL (SQL) via Sequelize ou Knex

```
const { Sequelize } = require('sequelize'); const sequelize = new Sequelize('database', 'user', 'password', { dialect: 'mysql' });
```

ORM/ODM populaires : Mongoose, Sequelize, Prisma

Gestion des promesses et asynchronisme

Callbacks : Méthode traditionnelle de gestion asynchrone

Promises : Gestion des opérations asynchrones

Async/Await : Syntaxe moderne pour écrire du code asynchrone

Gestion des erreurs en Node.js

Try...Catch : Gestion des erreurs synchrone et asynchrone

Middleware d'erreur dans Express

Process events : Capture des erreurs globales

Authentification et sécurité

JWT (JSON WEB TOKEN) : GÉNÉRATION DE JETONS D'AUTHENTIFICATION

CONST JWT = REQUIRE('JSONWEBTOKEN');

BCRYPT : HACHAGE DE MOTS DE PASSE

HELMET : PROTECTION DES EN-TÊTES HTTP

CORS (CROSS-ORIGIN RESOURCE SHARING) : GESTION DES ACCÈS INTER-ORIGINES

Tests avec Node.js

- **Jest** : Tests unitaires et d'intégration
  - test('addition', () => { expect(1 + 2).toBe(3); });

Optimisation des performances

Cluster : Exécution multi-thread pour utiliser les cœurs du CPU

Caching : Mise en cache des réponses avec Redis

Compression : Minification des réponses avec compression middleware

Outils de développement populaires

nodemon : Redémarrage automatique de l'application en cas de changement

dotenv : Gestion des variables d'environnement

Debugger : Outil intégré pour identifier les erreurs

- BDD

Introduction aux Bases de Données

Base de données (DB) : Collection de données organisées pour un accès, une gestion et une mise à jour efficaces.

SGBD (Système de Gestion de Base de Données) : Logiciel permettant de gérer les bases de données.

Exemples : MySQL, PostgreSQL, MongoDB, SQLite.

Types de bases de données :

Relationnelle (SQL) : Organisation en tables avec relations (ex. MySQL, PostgreSQL).

NoSQL (Non relationnelle) : Données sous forme de documents, colonnes, graphes, etc. (ex. MongoDB, Redis).

Différences entre SQL et NoSQL

Caractéristique	SQL (Relationnel)	NoSQL (Non relationnel)
Structure	Tables avec schéma fixe	Documents, colonnes, graphes
Langage	SQL	JSON, BSON, clé-valeur
Relations	Supportées	Non relationnelles
Scalabilité	Verticale (scale-up)	Horizontale (scale-out)
Exemples	MySQL, PostgreSQL	MongoDB, Redis

Étapes de conception d'une base de données relationnelle (SQL)

Analyse des besoins

Définition des entités et des relations entre elles.

Identification des données à stocker (clients, produits, commandes, etc.).

Conception conceptuelle (MCD - Modèle Conceptuel des Données)

Représentation des entités et des relations avec des diagrammes ERD (Entity-Relationship Diagram).

Exemple d'entités :

- Utilisateur (id, nom, email)
- Commande (id, date, utilisateur_id)

Conception logique (MLD - Modèle Logique des Données)

Conversion des entités en tables avec clés primaires et étrangères.

Normalisation pour éviter les redondances et assurer la cohérence des données (1FN, 2FN, 3FN).

Conception physique

Choix du SGBD (MySQL, PostgreSQL, SQL Server).

Indexation des colonnes fréquemment recherchées pour améliorer les performances.

Sécurité des données (droits d'accès, sauvegardes).

```
const mysql = require('mysql2');
const connection = mysql.createConnection({
  host: 'localhost',
  user: 'root',
  password: 'password',
  database: 'testdb'
});

connection.connect(err => {
  if (err) throw err;
  console.log('Connecté à MySQL');
});

connection.query('SELECT * FROM Utilisateur', (err, results) => {
  if (err) throw err;
  console.log(results);
});
```

Intégration de bases de données avec Node.js

ORM

Définition

L'ORM agit comme une **couche d'abstraction** entre l'application et la base de données, transformant les tables relationnelles en objets et vice versa. Cela permet aux développeurs d'interagir avec la base de données en utilisant des langages de programmation orientés objet, comme JavaScript, Python, ou Java, sans écrire directement des requêtes SQL.

Avantages de l’ORM

Abstraction des requêtes SQL : Plus besoin d'écrire du SQL brut, simplification du développement.

Productivité accrue : Manipulation des données via des objets et méthodes, plus intuitive pour les développeurs.

Sécurité : Réduction des risques d'injection SQL grâce à la gestion automatique des paramètres.

Compatibilité multi-SGBD : La même couche ORM peut fonctionner avec différentes bases de données (MySQL, PostgreSQL, SQLite, etc.).

Maintenance facile : Le code est plus lisible et plus simple à maintenir.

Inconvénients de l’ORM

Performance réduite : L'abstraction introduit une surcouche qui peut ralentir les opérations par rapport aux requêtes SQL optimisées manuellement.

Complexité pour les requêtes avancées : Les requêtes complexes peuvent être plus difficiles à écrire avec un ORM qu'en SQL pur.

Apprentissage : Nécessite d'apprendre l'outil et ses conventions propres.

ORM et ODM populaires pour Node.js

SQL : Sequelize, Knex.js

NoSQL : Mongoose, Firestore SDK

Avantages des ORM/ODM :

- Simplification des requêtes
- Validation des données
- Mapping entre objets JS et données

Sécurité et bonnes pratiques

- SQL
  - Protection contre l'injection SQL via des requêtes préparées.
  - Gestion des permissions utilisateur.
- NoSQL
  - Validation des données avant insertion.
  - Protection contre l'injection NoSQL (sanitize les entrées utilisateur).
- Sauvegarde et réplication : Backup régulier et répartition des données sur plusieurs serveurs.
- Utilisation d'outils de monitoring : pgAdmin pour PostgreSQL, MongoDB Compass.

Optimisation des performances

Indexation : Création d'index pour accélérer les recherches.

Partitionnement : Répartition des données sur plusieurs tables ou collections.

Mise en cache : Utilisation de Redis pour stocker temporairement des résultats.