Programmation Web – Avancé

BINV2150 A : JavaScript (& JAVA SERVLETS)

Week 1

R. Baroni / J.L. Collinet / C. Damas

0

Table des matières

Tous les sujets traités pendant ce cours...

Table des matières

Table des matières

1

Engagement pédagogique

Pour bien démarrer...

Généralités

- Unité d'enseignement :
  BINV2150-1 : Programmation Web – Avancé
- Activité d'apprentissage :
  BINV2150 A : JavaScript
- Nombre de crédits : 4

Organisation et durée

- Enseignement en Q1, examen en janvier
- 36h de pratique :
  - « Sessions A » Théorie et pratique de base : 12 X 2h (semaines 1 à 12) avec 2 enseignants / session
  - « Sessions B » Pratique approfondie : 6 X 2h (semaines 7 à 12) avec 1 enseignant / session

Organisation et durée

- 3 enseignants :
  - Raphael Baroni : « Sessions A » pour 3 groupes, « Sessions B » pour 1 groupe
  - Jean-Luc Collinet : « Sessions A » pour 1 groupe, « Sessions B » pour 2 groupes
  - Christophe Damas : « Sessions A » pour 2 groupes

Articulation avec le cursus

- Le cours est placé au bloc 2.
- Prérequis pour accéder à ce cours :
  - BINV1020-1 - Analyse et programmation orientée objet
  - BINV1030-1 - Gestion de données : bases
  - BINV1050-1 - Développement web : bases
  - BINV1110-1 - Projet de développement web

Articulation avec le cursus

- Corequis pour accéder à ce cours
  - BINV2160-1 – Analyse et modélisation
- Cours dont cette activité d'apprentissage est un corequis:
  - BINV2090-1: Conception d'applications d'entreprise

Articulation avec le cursus

- Cours dont cette activité d'apprentissage est un prérequis :
  - BINV3080-1 : Intégration en milieu professionnel
  - BINV3140-1 : Programmation : questions spéciales
  - BINV3150-1 : Développement web : questions spéciales, Développement Web : questions spéciales

Compétences de formation développées

- A la fin du cours, l'étudiant·e sera capable de :
  - Réaliser une animation en JS.
  - Communiquer des données de manière synchrone ou asynchrone entre une application cliente HTML/CSS/JS et un backend JAVA par le biais de messages JSON.
  - Authentifier un utilisateur de manière sécurisée via JWT et gérer des cookies.

Acquis d'apprentissage spécifiques

- Elaborer une animation en JS directement via les API du browser.
- Elaborer une animation via une librairie JS.
- Créer un backend Java qui répondent à des requêtes synchrones ou asynchrones d'un frontend HTML/JS/CSS & Bootstrap.
- Appliquer JQuery pour optimiser le frontend.

Acquis d'apprentissage spécifiques

- Développer une single-page Web application communiquant avec un backend Java par le biais de messages structurés en JSON.
- Transformer un JSON en un objet Java, et inversement.
- Utiliser les JSON Web Token et les cookies pour authentifier un utilisateur.

Présentation du contenu de l'activité d'apprentissage

- Introduction au contexte d'utilisation de JS
- Introduction au langage JS côté client, à l'utilisation d'APIs du navigateur et de librairies JS
- Introduction aux communications (synchrones) client /serveur

Présentation du contenu de l'activité d'apprentissage

- Introduction aux single-page web applications et aux communications asynchrones client / serveur
- Introduction à l'authentification sécurisée d'un utilisateur et à la gestion de cookies

Méthode et mode d'enseignement

- Une séance commence généralement par des notions théoriques et des démonstrations.
- Elle se poursuit par des exercices pratiques.
- Vers la fin du quadrimestre, un projet est à réaliser en classe et à la maison SSI nécessaire.
- Pour pouvoir suivre le cours, l'étudiant a besoin des supports et des outils informatiques listés plus loin.

Modalités et critères d'évaluation

ID	Compétence	Eval 1ère session	Eval 2ème session
1	Réaliser une animation en JS	Projet (50%) Examen (50%)	Examen (100%)
2	Communiquer des données de manière asynchrone entre une application cliente HTML/CSS/JS et un backend JAVA		
3	Authentifier un utilisateur de manière sécurisée et gestion des cookies		

Modalités et critères d'évaluation

- Les compétences de l'étudiant·e seront évaluées par le biais d'un projet (50% des points) et d'un examen (50% des points).
- Le projet consiste à la mise en œuvre d'un programme reprenant les technologies vues en cours et appliquant des librairies JS à explorer par l'étudiant·e.

Modalités et critères d'évaluation

- L'examen consiste en la réalisation de programme(s) reprenant les compétences développées lors du cours.
- En deuxième session, les compétence de l'étudiant·e seront évaluées par le biais d'un examen seulement (100% des points). Toute évaluation réalisée préalablement ne sera pas prise en compte dans l'évaluation finale.

Supports de cours et bibliographie

- Cette présentation [JS-XY-BINV2150-A]
- Moodle : https://moodle.vinci.be/
- JavaScript Guide (EN) [2] : https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide

Supports de cours et bibliographie

- Tutoriels légers pour apprendre le JS (EN) [3] & [4] : https://www.w3schools.com/js/default.asp, https://www.tutorialspoint.com/javascript/index.htm
- Toute la biographie (voir fin de la presentation)

Prise de contact avec les enseignants

- 3 enseignants : Raphael Baroni, Jean-Luc Collinet, Christophe Damas
- Prioritairement pendant les cours
- Via le forum sur Moodle pour des sujets utiles à la communauté
- Via email pour des sujets personnels

2

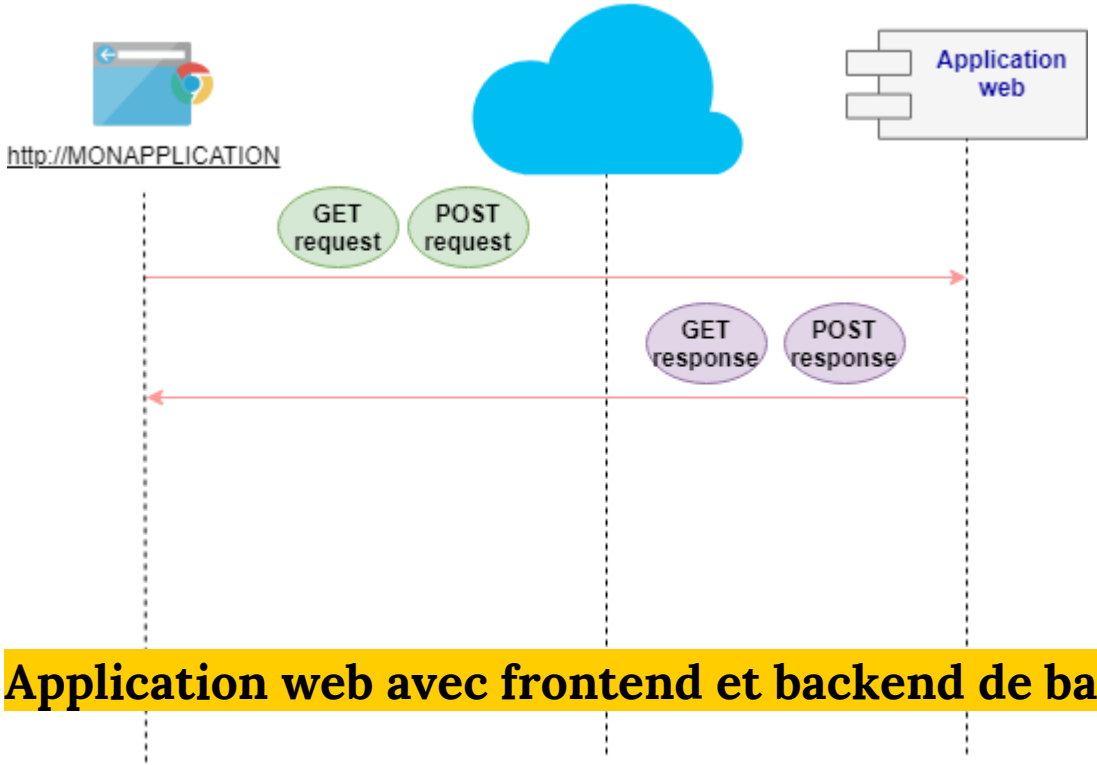
Introduction au context d'utilisation de JS

Pourquoi le JS ? Où est-ce que ça intervient ? ...

Introduction

- But du JS ?
- JS dans quel contexte ?
- Permettre des interactions dynamiques avec l'HTML et le CSS d'un browser.
- Exécution de JS après la génération d'une page web comprenant du HTML et potentiellement du CSS.

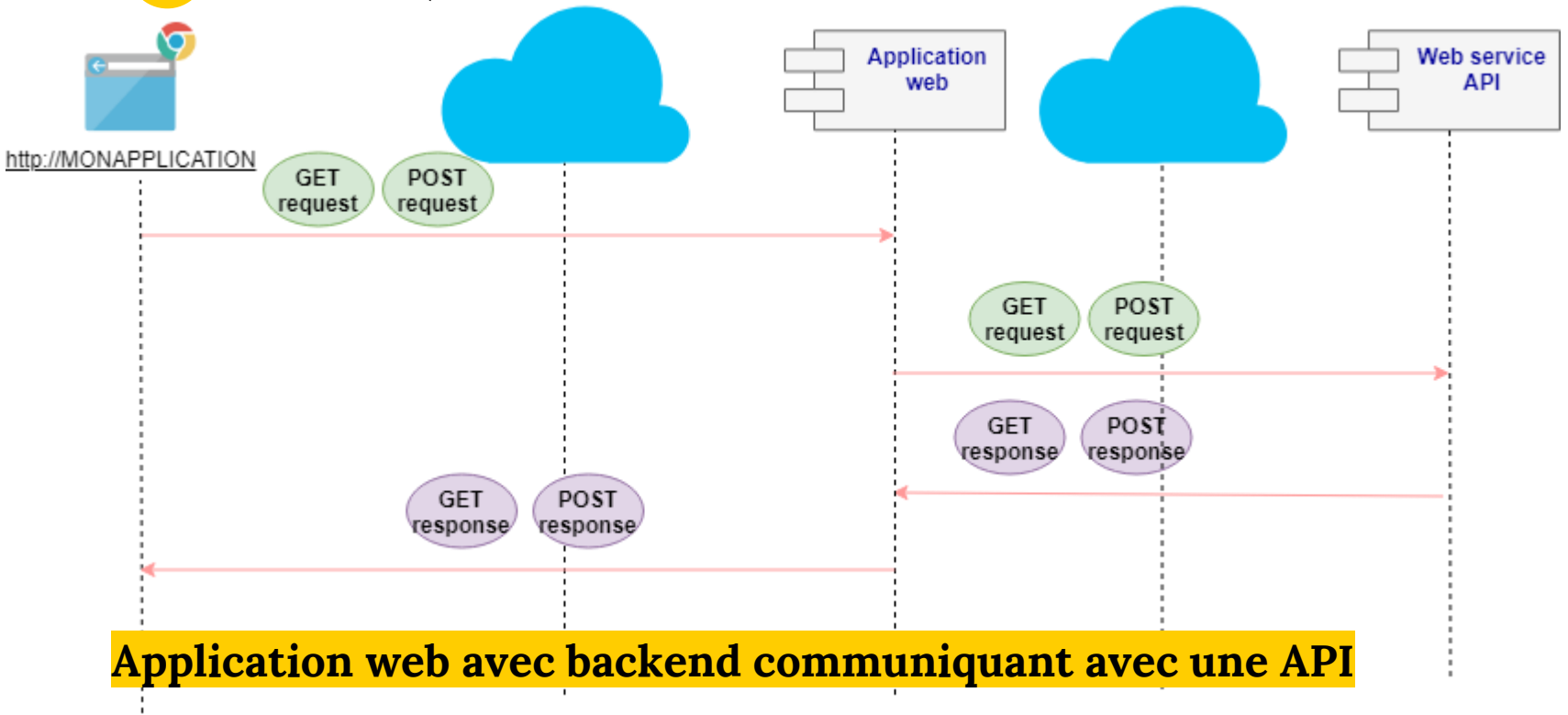
Différents contextes d'applications clients / serveurs



- Synchrone ?
- Asynchrone ?

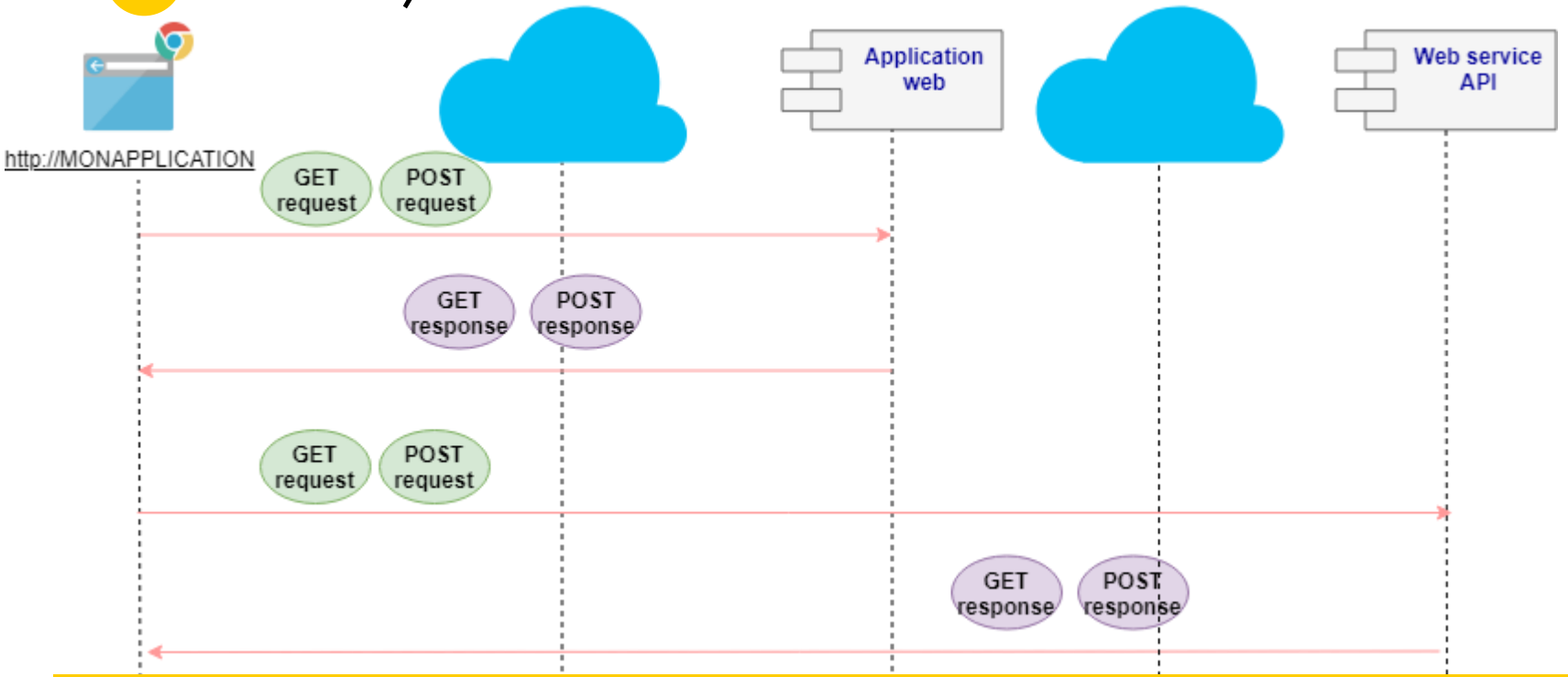
Application web avec frontend et backend de base

Différents contextes d'applications clients / serveurs



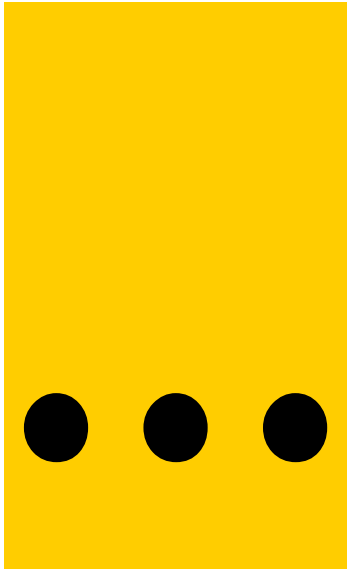
Application web avec backend communiquant avec une API

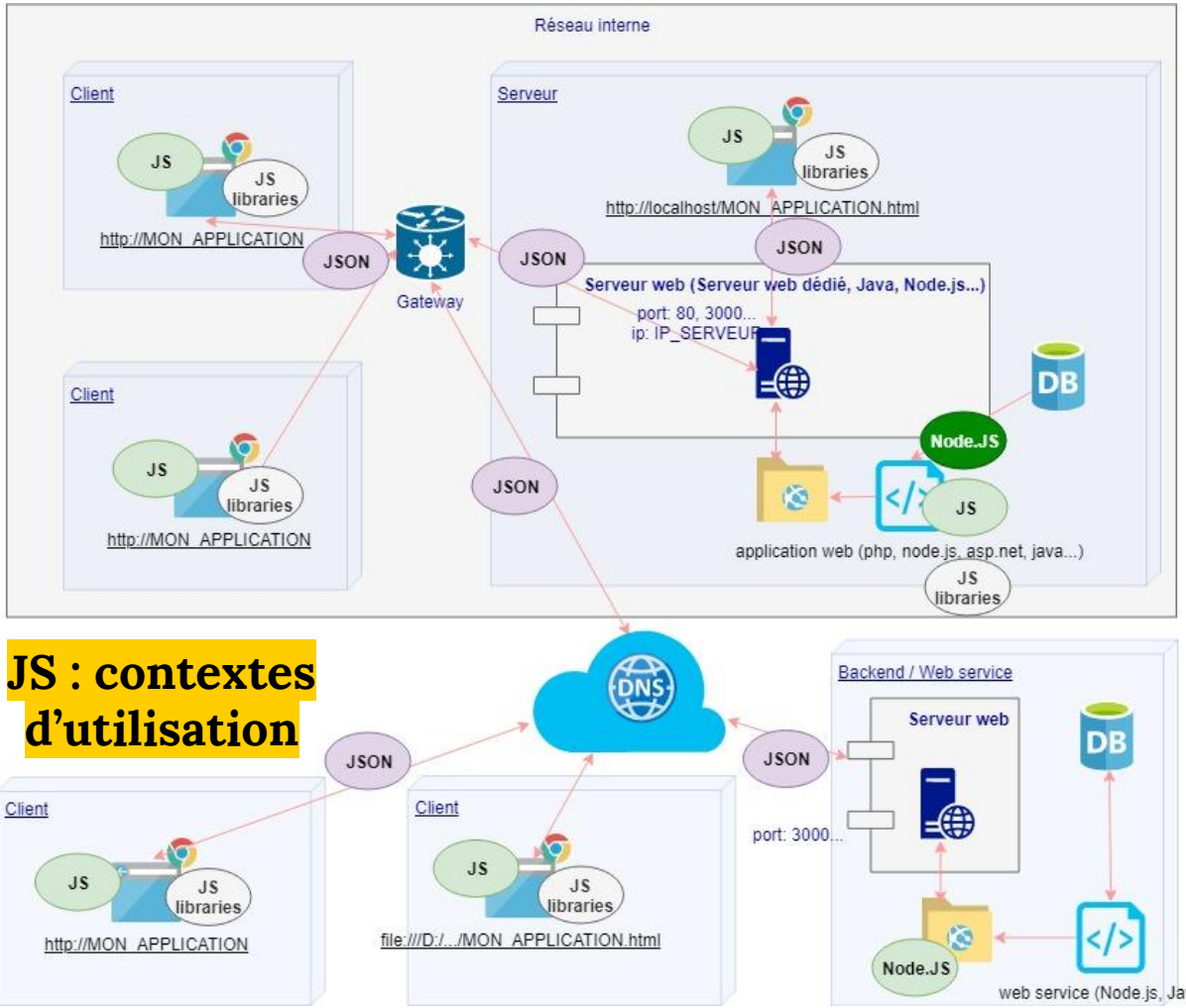
Différents contextes d'applications clients / serveurs



Application web avec frontend communiquant avec backend de base & API

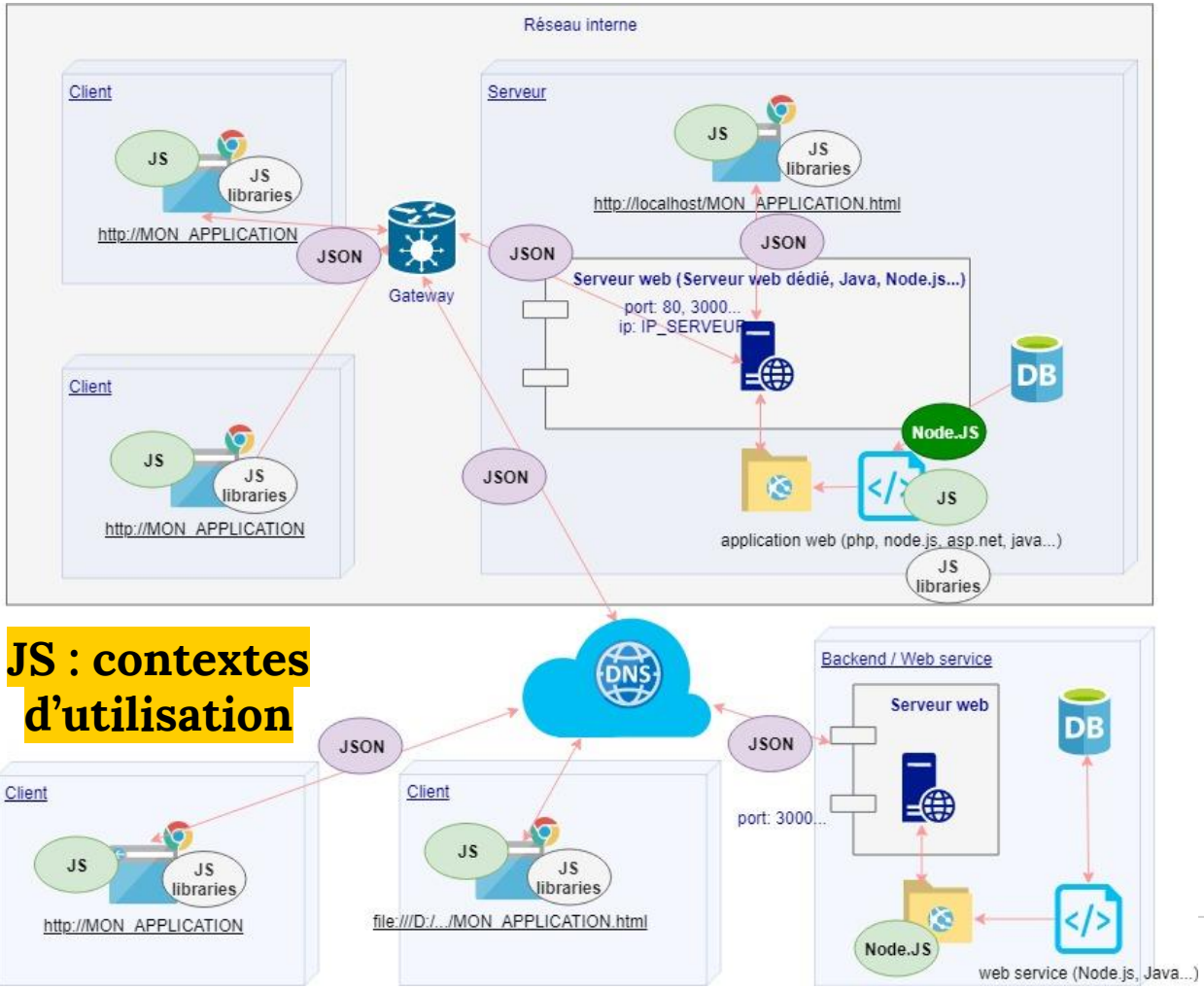
Différents contextes d'applications clients / serveurs





1. Client

- Programmation JS et utilisation des APIS du browser (API DOM...)
- Librairies JS & frameworks : JQuery, Phaser, Anime.js, AngularJS...

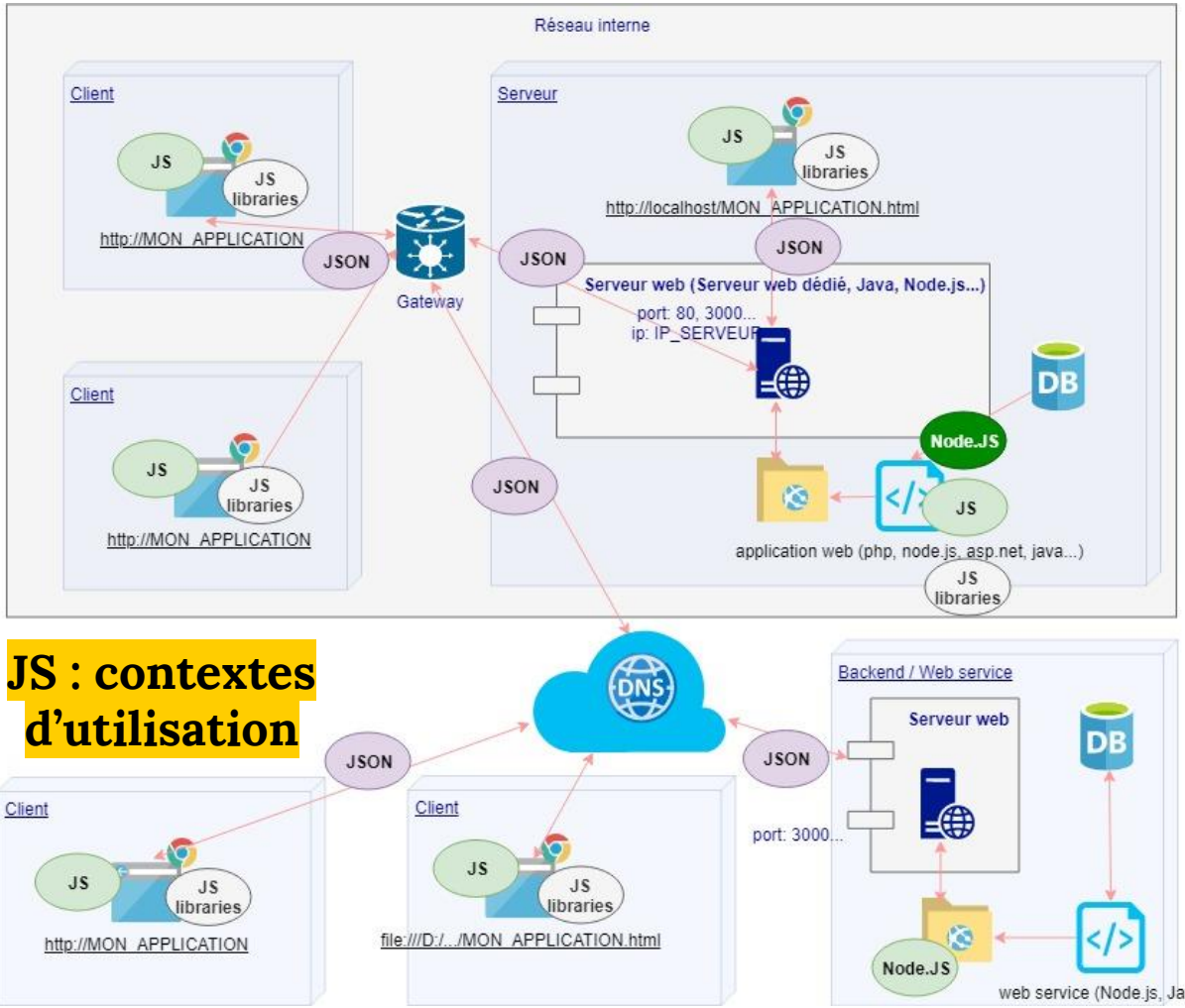


2. Serveur

- Node.js
- Express
- Sail
- ...

3. Communication des données

- JavaScript Object Notation (JSON)



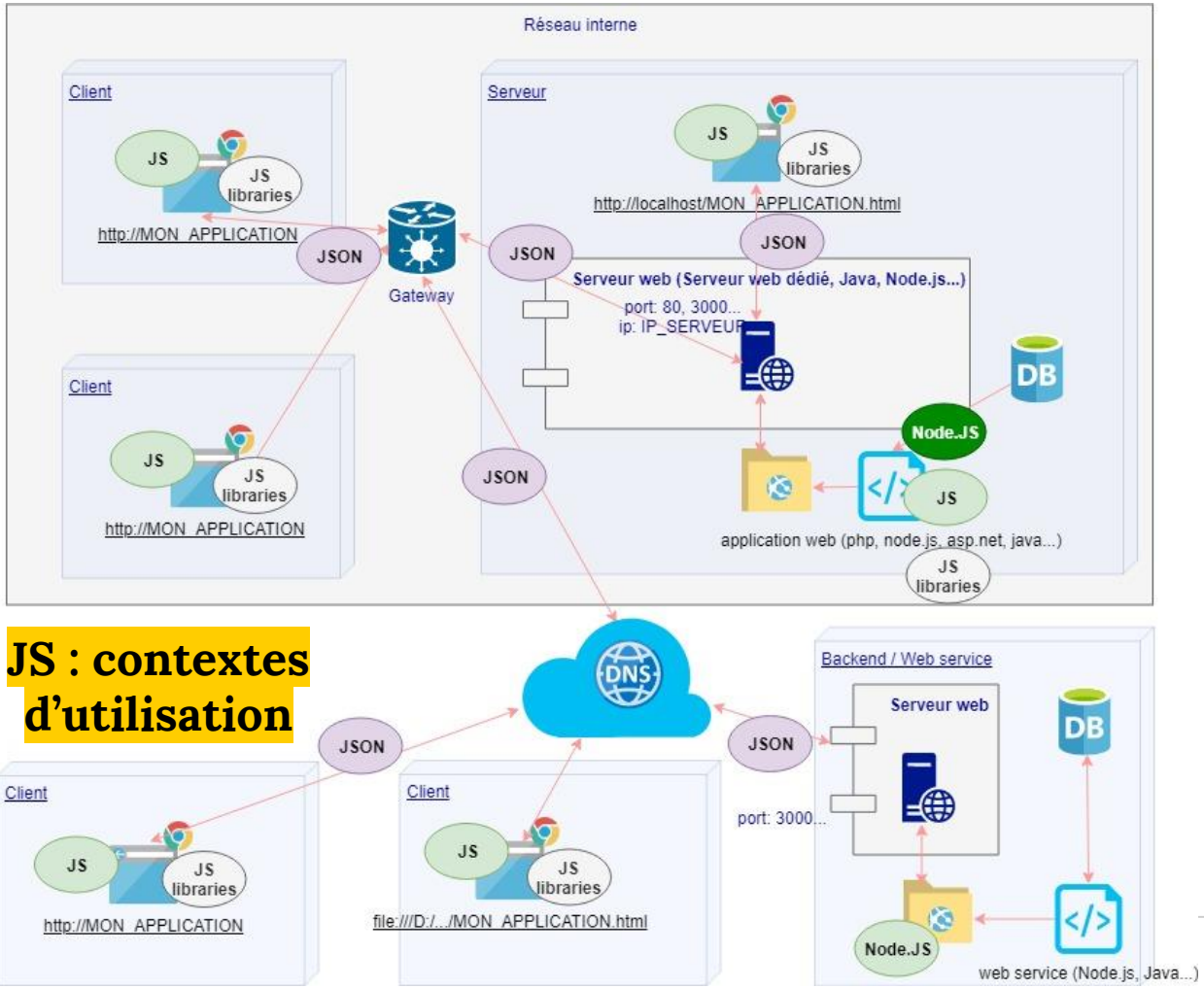
JS : contextes d'utilisation

4. Sur-ensemble de JS / langage compliant vers JS :

- TypeScript
- CoffeScript
- ...

5. Scripts embarqués

- Chrome extensions
- Google Apps Script
- ...



JS : contextes d'utilisation

6. Applications multi-plateformes de bureau

- https://electronjs.org/
- https://cordova.apache.org/

...

3

Introduction au langage JS côté client, à l'utilisation d'APIs du navigateur et de librairies JS

Découvrons le langage côté client...

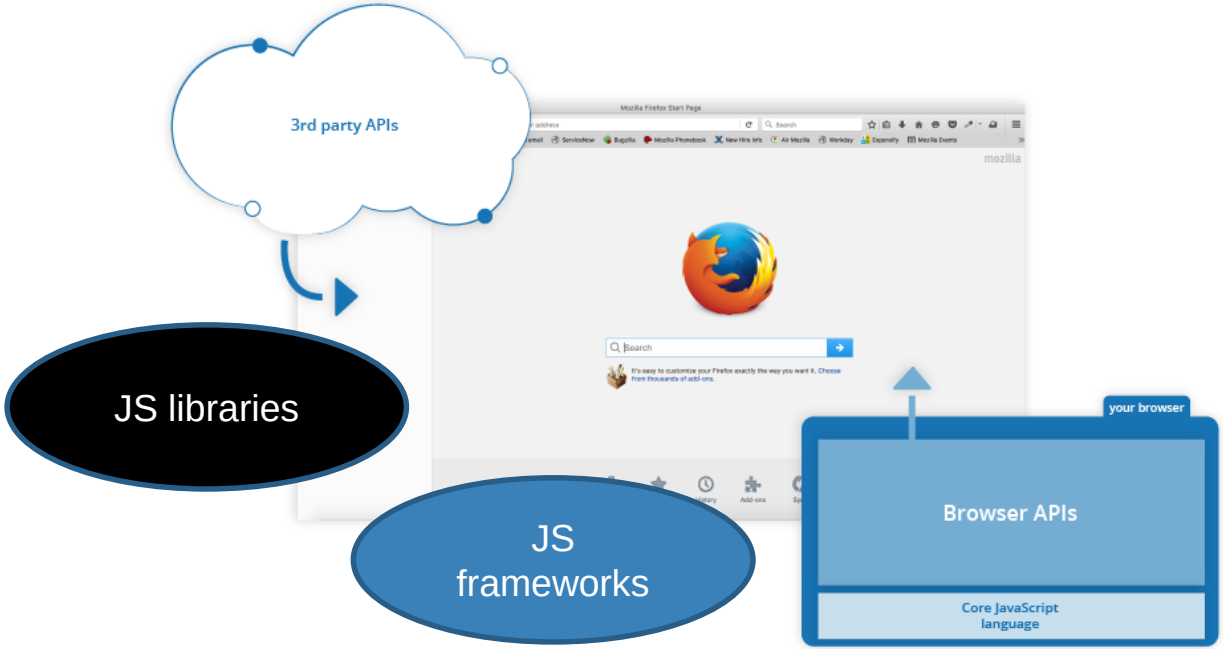
Table des matières :
3. Introduction au langage JS côté client, à l'utilisation d'APIs du navigateur et de librairies JS

Table des matières :
3. Introduction au langage JS côté client, à l'utilisation d'APIs du navigateur et de librairies JS

Table des matières :
3. Introduction au langage JS côté client, à l'utilisation d'APIs du navigateur et de librairies JS

Table des matières :
3. Introduction au langage JS côté client, à l'utilisation d'APIs du navigateur et de librairies JS

Introduction aux JS côté-client



Relation entre APIs, le browser et le JS [1]

Interaction de base avec ou sans un browser : Quels programmes utiliser ?

- Browser : pour le cours, **Chrome** sera privilégié
- Pour développer les scripts : **Visual Studio Code** & **Visual Studio Live Share** seront privilégiés
- Pour du code JS pur (sans HTML) ou pour gérer des packages : **Node.js** ou **npm**

Interaction de base avec ou sans un browser : où mettre le code ?

A. Directement dans un browser
   - Accès à la console : **F12** ou **CTRL + SHIFT + i** (Chrome)
   - **DEMO** : opérations mathématiques de base dans Chrome

B. Directement via l'interpréteur de commandes de Node au sein d'un terminal : **node**
   - Pourquoi Node.js ?
   - **DEMO** : vérifier l'installation de Node.js & opérations mathématiques de base

Interaction de base avec ou sans un browser : où mettre le code ?

C. Via l'interpréteur de commandes de Node au sein d'un terminal et d'un script externe : **node script_name** (**.js** est optionnel)
   - **DEMO-00** : opérations mathématiques de base dont le résultat est affiché dans le terminal au lancement du script

Interaction de base avec ou sans un browser : où mettre le code ?

D. Dans une page HTML

1. Dans une fonction ou événement associé à un élément HTML
   - **DEMO-01A** : affichage d'une alerte au chargement d'une page
   - Notion : **<body onload='function_Name()'>**

Interaction de base avec ou sans un browser : où mettre le code ?

D. Dans une page HTML

2. Dans la section **<body>** ou **<head>** d'un fichier HTML, à l'aide de la balise **<script>**
   - **DEMO-01B** : même fonctionnalité

Interaction de base avec ou sans un browser : où mettre le code ?

D. Dans une page HTML

3. Dans un fichier externe avec **<script src='file_name.js'>** dans un fichier HTML à la fin de la balise <body>
   - **DEMO-01C** : même fonctionnalité

Instructions JS

- Composition d'une instruction JS (« statement » normalement séparé par « ; ») à exécuter par le browser :
  - Expressions
  - Valeurs
  - Opérateurs (+, -...)
  - Mots clés (for, break...)
  - Commentaires.

Instructions JS

- **Bonne pratique** : séparer chaque instruction par un « ; ».

```
let x = 1 ;
console.log(‘x = ‘, x);
```

Les commentaires

- Ajout de commentaires : via // ou /* et */

```
function raiseAlert() {
    // Affichage d'un message via un pop-up.
    alert("This is an alert.");
    /* Affichage d'un message
       dans la console du browser.
    */
    console.log("An alert has been raised.");
}
```

Déclaration, initialisation et mise à jour de variables

- Variables sensibles à la casse

```
let monBrowser ;
//est différent de
let monbrowser ;
```

- Pas de déclaration du type de variable (« langage dynamiquement typé »)
  - Type d'une variable déterminé à l'exécution

Déclaration, initialisation et mise à jour de variables

- Pour une portée associée à un bloc : **let**
  - variable pas accessible en dehors du block, processée à l'exécution seulement
  - pas de redéclaration possible dans le même bloc
  - Pas de création de propriété sur l'objet global (this.x)
  - Détails : https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/let

Déclaration, initialisation et mise à jour de variables

- Pour une portée associée à un bloc mais immutable : **const**
- Pour une portée associée à une fonction ou portée globale si déclaration en dehors de toute fonction : **var**
  - processée avant l'execution du code ("hoisting") ;
  - variable "globale" accessible au travers de tout le programme ;
  - redéclarable dans n'importe quel bloc.

Déclaration, initialisation et mise à jour de variables

- Pour l'assignation d'une valeur à une variable non déclarée :
  - Création implicite d'une variable globale

```
i=10;
```

Déclaration, initialisation et mise à jour de variables

- Types de variables
  - **Number** (Nombre) : un seul type pour les entiers, réels, doubles...
  - **String** (Chaîne) : une chaîne est comprise entre guillemets simples ou doubles. « Echapper un caractère » en mettant un backslash ****
  - **Bool** (Booléen)
  - **Array** (Tableau)
  - **Object** (Objet)

Déclaration, initialisation et mise à jour de variables

- Renvoi sous forme d'une chaîne le type d'une expression : opérateur **typeof**
- **DEMO-1D** : Utilisation de la console d'un browser afin de déclarer une variable, d'y assigner une valeur de type « string », de vérifier le type de donnée placé dans la variable. D'assigner un entier ou un réel, et de vérifier le type de donnée.

Exercices

A vous de jouer...

- **EX-00A** : Créer une page HTML affichant, au chargement, un prompt (prompt()) demandant un nom. Puis affichez un message d'alerte ayant été préalablement défini dans une variable, sur base du résultat du prompt. Veillez à ce que le JS se trouve dans un script externe.
- **EX-00B** : Affichez dans votre message d'alerte la date et l'heure, dans un format agréable à lire.

- **EX-00C (optionnel)** : Créez un jeu demandant de deviner un mot. En cas de succès, félicitez le joueur, puis redirigez le vers un site de votre choix. En cas d'erreur, donner un indice dans votre prompt pour aider le joueur. Après 5 tentatives ratées, affichez un message d'échec.
  Demandez à votre voisin d'y accéder et de jouer.

Références

[1]	MDN web docs, Introduction to web APIs. Lien : https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Client-side_web_APIs/Introduction
[2]	MDN web docs, JavaScript Guide. Lien : https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide
[3]	w3schools.com, JavaScript Tutorial. Lien : https://www.w3schools.com/js/default.asp
[4]	tutorialspoints.com, Javascript Tutorial : Lien : https://www.tutorialspoint.com/javascript/index.htm