FORMATION JQUERY

Openska

QU'ATTENDEZ-VOUS DE LA FORMATION ?

Qui êtes vous ?

Quel est votre profil ?

Où travaillez vous ?

Quel est votre parcours professionnel ?

Qu'attendez vous de la formation ?

DÉROULEMENT D'UNE JOURNÉE

09:30 - 11:00

Pause

11:15 - 12:45

Pause déjeuner

14:00 - 15:15

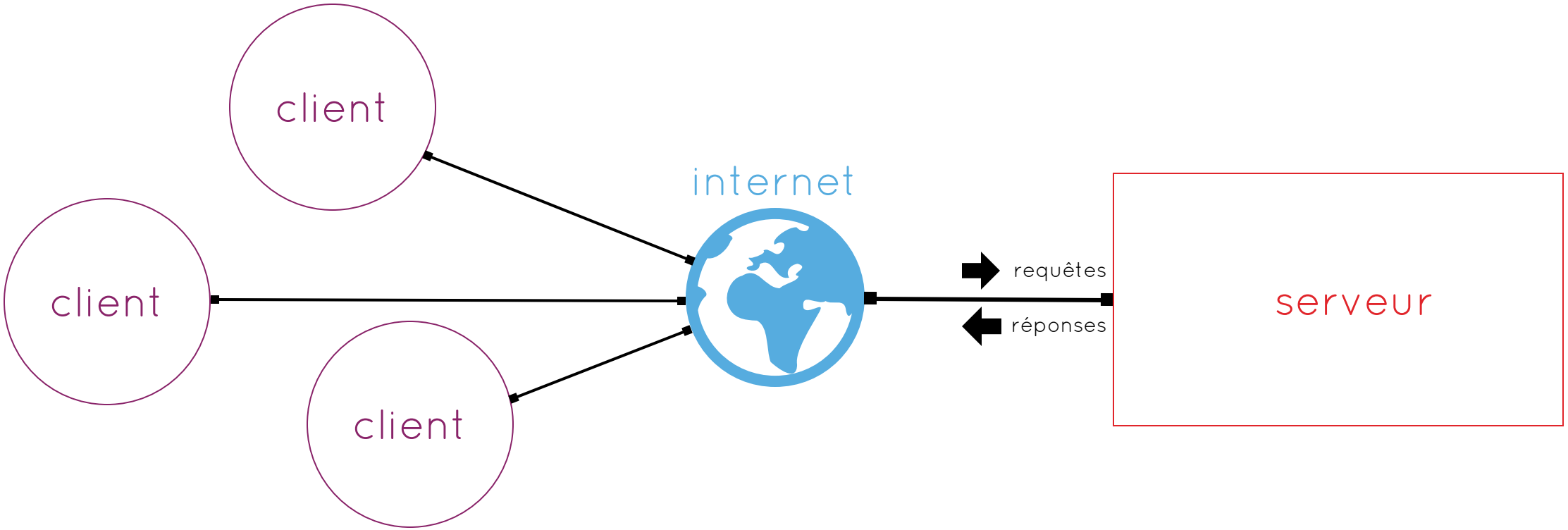
Pause

15:30 - 17:00

SOMMAIRE

GÉNÉRALITÉS JAVASCRIPT

CLIENT / SERVEUR



LES TECHNOLOGIES DU WEB

Coté SERVEUR

PHP	Symfony
JAVASCRIPT	NodeJS
GO	Iris
...	

Coté CLIENT

HTML	
JAVASCRIPT	jQuery
	Angular / React
...	

JAVASCRIPT ET HTML

```
<!DOCTYPE html>
<html>
    <head>
        <title>Openska</title>
    </head>

    <body>
        <h1>Openska</h1>

        <script type="text/javascript">
            let a = 5;
            let b = 5 + a;

            console.log('Hello World !');
        </script>
    </body>
</html>
```

ÉVOLUTION

Le Javascript est un langage dynamique, qui évolue énormément

Il peut être compliqué de savoir quelles sont les nouveautés acceptées par les navigateurs, celles qui le seront et celles qui ne le sont pas

http://caniuse.com/

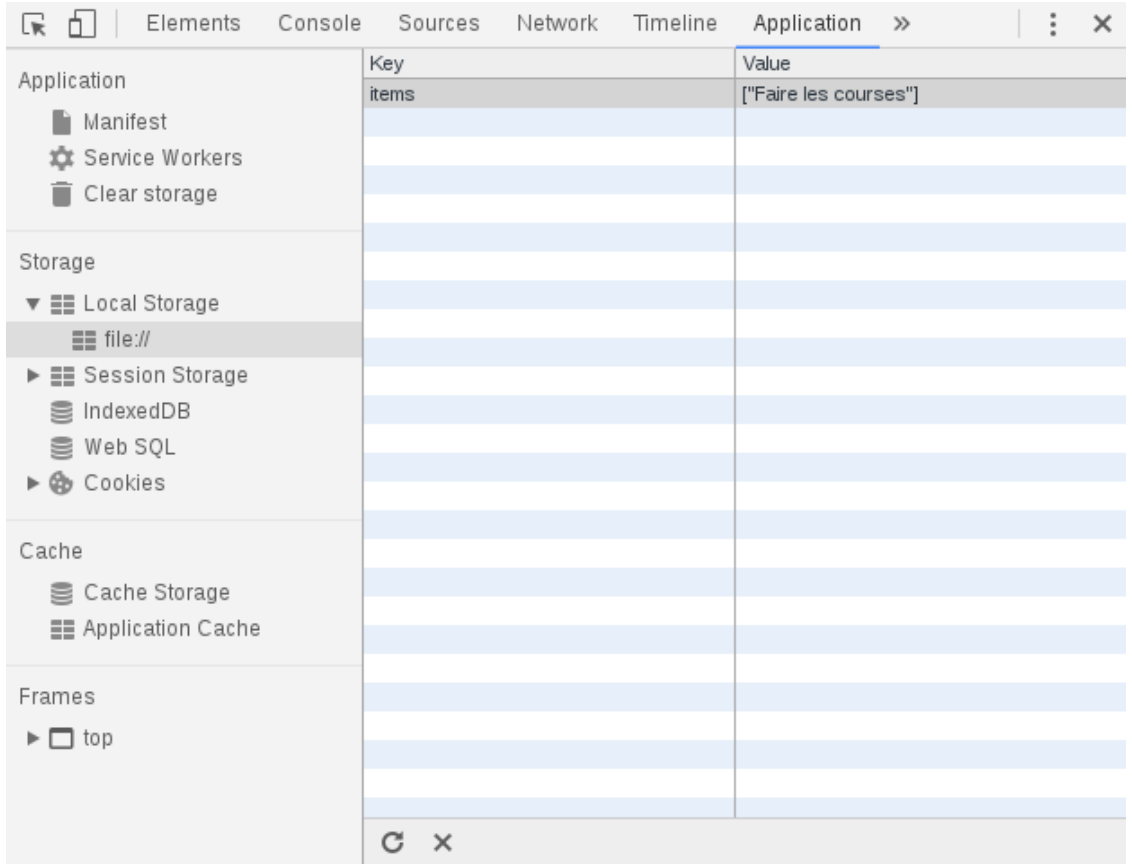
API HTML 5

Il existe aujourd'hui des fonctionnalités natives qui peuvent être très utile, surtout depuis l'apparition du développement mobile

On peut utiliser en javascript des outils comme la géolocalisation, les notifications, le stockage local, ...

EXEMPLE AVEC LE STOCKAGE LOCAL

```
// set.html
let items = ["Faire les courses"];
let json = JSON.stringify(items);
localStorage.setItem('items', json)


// get.html
let json = localStorage.getItem('items');
let items = JSON.parse(json);
console.log(items);
```



INTRODUCTION À JQUERY

LE FRAMEWORK JQUERY

Créer en janvier 2006, jQuery est open source, financé par des dons et maintenu par la communauté

L'objectif de jQuery : Simplifier et standardiser les développement javascript

Largement utilisé et documenté : https://jquery.com/

POURQUOI JQUERY

Léger, seulement 32Kb (minifié et gzippé)

Compatible CSS3, permet d'utiliser les sélecteurs CSS3 pour sélectionner des éléments

Cross-Browser, Chrome, Firefox, Edge, IOS, Android, ...

JQUERY ET $

"$" est un objet créer par jQuery, et de type jQuery

"$" est est raccourci pour l'objet jQuery

Toutes les fonctions de jQuery retournent l'instance courante de l'objet sur lequel elles travaillent, son usage est donc chainable

Attention, l'objet "$" peut être défini par d'autres librairies !

$ ET CONFLICTS

```
// Aucun conflict si jquery est la seule librairie
$("body").append("<p>Hello World!</p>");


// Utiliser la fonction noConflict() de jQuery
var $jq = jQuery.noConflict(true);
$jq("body").append("<p>Hello World!</p>");


// Utiliser le scope d'une fonction
(function($){
    $("body").append("<p>Hello World!</p>");
})(jQuery);
```

LE DOM
ET SA MANIPULATION

PRINCIPE DU DOM

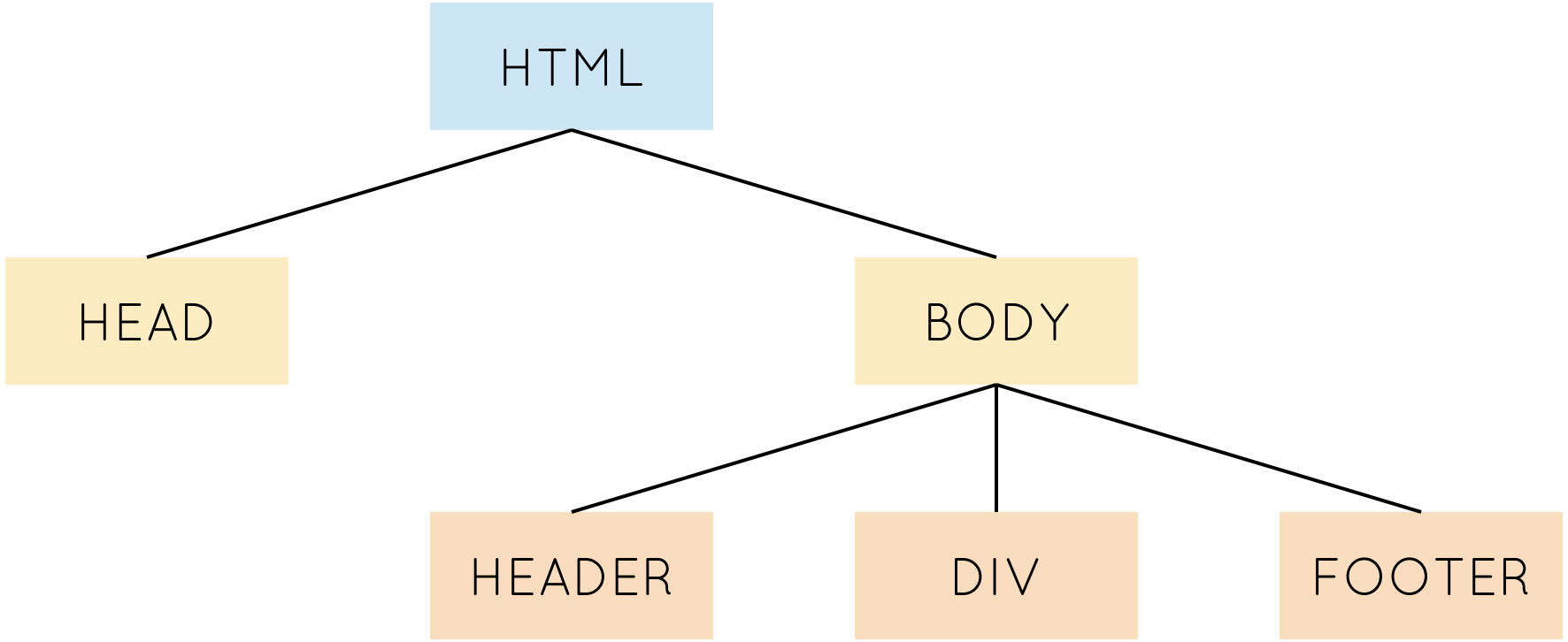
DOM : Document Object Model

Maintenu par W3C

Permet de gérer les éléments d'un documents en suivant une certaine norme

Un document expose un arbre d'éléments pouvant avoir des attributs

STRUCTURE EN ARBRE



LES SÉLECTEURS CSS

Avec jQuery, on retrouve les sélecteurs CSS3 qu'on a l'habitude d'utiliser

On utilisera "#" pour sélectionner un ID et "." pour sélectionner une classe

On utilisera l'espace pour sélectionner des descendants et des virgules pour utiliser plusieurs sélecteurs

```
$('#mon-id');

$('.ma-classe');

$('#mon-id .ma-classe');
```

EXEMPLES DE SÉLECTIONS

```
/* JAVASCRIPT */
let id = document.getElementById('mon-id');
let querySelectorId = document.querySelector('#mon-id');
let classes = document.getElementsByClassName('ma-class');
let querySelectorClasses = document.querySelectorAll('.ma-class');
let div = document.getElementsByTagName('DIV');
let inputs = document.querySelectorAll('input[type="text"]');


/* JQUERY */
let id = $('#mon-id');
let classes = $('.classes');
let div = $('div');
let inputs = $('input[type="text"]');
```

MODIFICATION DU DOM

Le DOM est accessible en écriture

On peut créer des éléments

... et les attacher à un noeud existant

On peut modifier des évènements, ou en créer

On peut modifier des noeuds, leurs attributs ou leurs contenus

CRÉATION D'UN ÉLÉMENT

```
/* JAVASCRIPT */
let el = document.createElement('h2');
el.textContent = 'Hello !';
document.getElementById('header').appendChild(el);


/* JAVASCRIPT + JQUERY */
$('#header').append($('<h2>').text('Hello !'));
```

LECTURE D'UN ÉLÉMENT

```
let href = $('#mon-id').attr('href');
let dataId = $('#mon-id').data('id');
let content = $('#mon-id').text();
let htmlContent = $('#mon-id').html();
let tag = $('#mon-id').prop('tagName');

console.log(href, dataId, content, htmlContent, tag);
```

ACCÈS AUX PARENTS, ENFANTS, FRÈRES

```
let elem = $('#mon-id')

// élément parent direct
let parent = elem.parent();

// éléments parents
let parents = elem.parents();

// éléments frères (au même niveau)
let siblings = elem.siblings();


// éléments enfants directs
let children = eleme.children();
```

MODIFICATIONS

```
let elem = $('#mon-id');

elem.attr('href', '#updated');
elem.data('id', 7628);
elem.text('Hello World');
elem.html('<h1>Hello World !</h1>');

// Move élement
elem.appendTo($('destination'));
elem.prependTo($('destination'));
```

LES ÉVÉNEMENTS

PRINCIPES DES ÉVÉNEMENTS

DOM décrit aussi les interactions de l'utilisateur avec la structure ou le contenu de notre arbre

A chaque action de l'utilisateur sur un élément de l'arbre, correspond un événement

On peut attacher une fonction à un événement

ÉVÉNEMENTS LIER À LA PAGE / FENÊTRE

onabort : interruption du chargement de la page

onload : après la fin du chargement de la page

onbeforeunload : juste avant de quitter la page

...

ÉVÉNEMENTS LIER À LA SOURIS

onclick : sur un simple clic

ondblclick : sur un double clic

onmousemove : lorsque la souris est déplacée

onscroll : lorsque le scroll de la souris est utilisé

...

ÉVÉNEMENTS LIÉS AU CLAVIER

onkeydown : à l'enfoncement d'une touche

onkeypress : lorsqu'une touche est pressée puis relâchée

onkeyup : au relâchement d'une touche

...

ÉVÉNEMENTS LIER AUX FORMULAIRES

onblur : à la perte du focus

onfocus : lorsque qu'un élément est ciblé

onsubmit : à la soumission d'un formulaire

...

MANIPULATION D'ÉVÉNEMENTS

```
/* ÉVÉNEMENT SUR UN ÉLÉMENT */
$(document).on('click', '#add', event => console.log(event));


/* ÉVÉNEMENT SUR UN ÉLÉMENT EXISTANT */
$('#add').on('click', event => console.log(event));


/* ÉQUIVALLENT AU PRÉCÉDENT */
$('#add').click(event => console.log(event));
```

CRÉER SES PROPRES ÉVÉNEMENTS

```
setTimeout(() => {
    /* ÉVÉNEMENT PERSONNALISÉ + ARUMENTS */
    $('body').trigger('notif-alert', ['Connexion impossible.']);
}, 2500);

$('body').on('notif-alert', (event, message) => {
    let notif = $('<div>').addClass('notif').addClass('notif-alert').text(message);
    $(event.target).append(notif);
});

$('.btn-send').on('click', event => {
    console.log('click');
    $(event.target).prop('disabled', true);
    setTimeout(() => {
        /* ÉVÉNEMENT PERSONNALISÉ */
        $('.btn-send').trigger('sent');
    }, 2500);
});

$('.btn-send').on('sent', event => {
    $(event.target).prop('disabled', false);
});
```

Lancement d'une notification

Désactive temporairement un bouton, le temps de finir la fin d'une action

AJAX

INTRODUCTION À AJAX

Asynchronous JAvascript and XML

Permet de faire des requêtes asynchrones, communiquer avec le serveur sans recharger entièrement une page web

Permet de diminuer le temps de latence, et d'augmenter la réactivité des applications

REQUEST VS $.AJAX

Il existe plusieurs moyens de faire des requêtes AJAX

XMLHttpRequest : Plus ancien et moins naturel, à l'avantage d'être du Javascript natif

Request / fetch : Plus récent et plus naturel, permet de faire des requêtes asynchrones, nativement en utilisant les promesses

$.ajax : Plus ancien que le précédent, il a l'avantage d'être cross-browser, d'être stable et d'avoir fait ses preuves

AJAX AVEC REQUEST

```
let myGetRequest = new Request('https://jsonplaceholder.typicode.com/posts');
fetch(myGetRequest)
    .then(response => {
        response.json().then(data => console.log(data))
    });
```

```
let myPostHeaders = new Headers();
myPostHeaders.append('Content-Type', 'application/json');
let options = {
    method: 'POST',
    headers: myPostHeaders,
    body: JSON.stringify({userId: 1, title: 'Formation Openska', body: 'Lorem ipsum dolor sit amet ...'}),
};

let myPostRequest = new Request('https://jsonplaceholder.typicode.com/posts');
fetch(myPostRequest, options)
    .then(response => {
        response.json().then(data => console.log(data));
    });
```

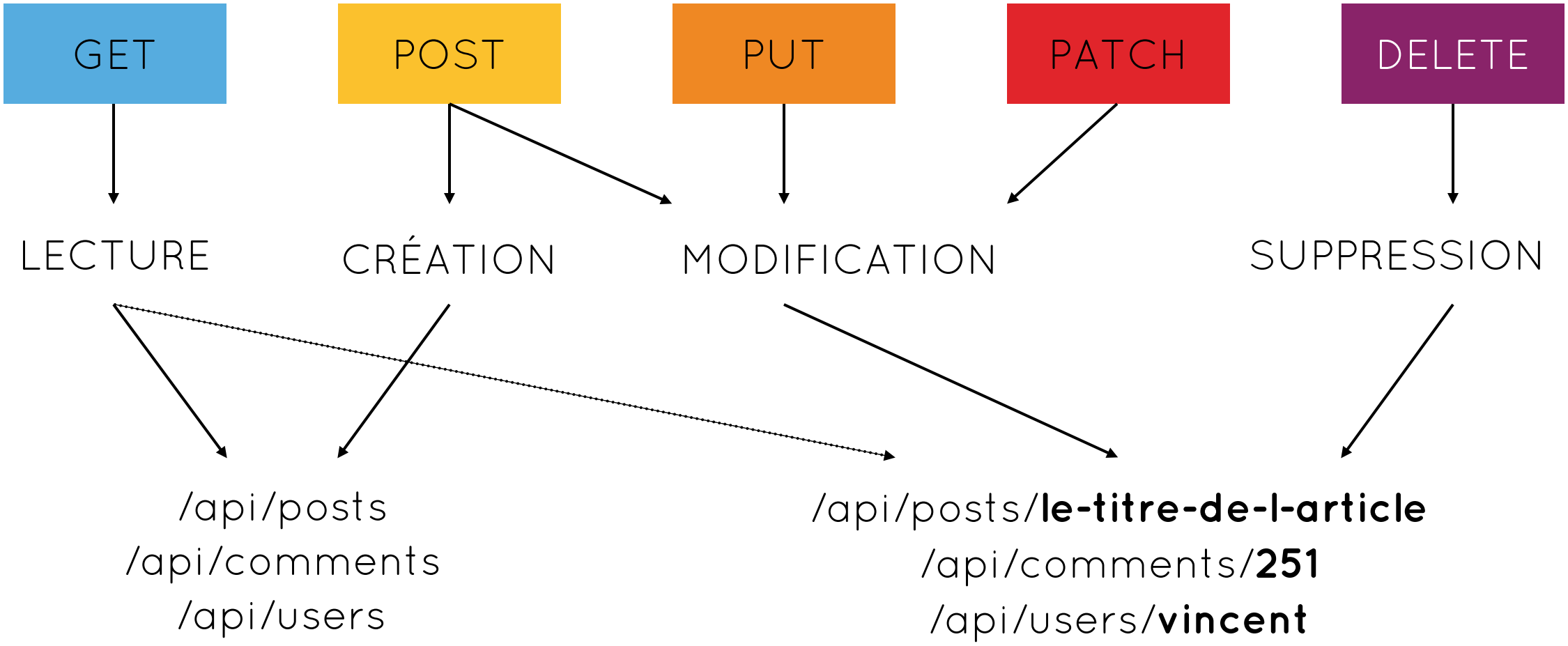
AJAX AVEC $.AJAX

```
$.ajax('https://jsonplaceholder.typicode.com/posts').done(data => console.log(data));


let data = {userId: 1, title: 'Formation Openska', body: 'Lorem ipsum dolor sit amet ...'};
$.ajax({
    url: 'https://jsonplaceholder.typicode.com/posts',
    method: 'POST',
    data: data,
}).done(data => console.log(data));
```

AJAX AVEC $.SHORTCUT

```
$.get('https://jsonplaceholder.typicode.com/posts', data => console.log(data));


let data = {userId: 1, title: 'Formation Openska', body: 'Lorem ipsum dolor sit amet ...'};
$.post('https://jsonplaceholder.typicode.com/posts', data, data => console.log(data));
```

UTILISER UNE API REST



CE N'EST PAS TOUT !

La réponse elle aussi utilise le protocole HTTP

Les codes HTTP : 100, 200, 300, 400 et 500

Hypermédia ...

http://putaindecode.io/fr/articles/api/hateoas/

ANIMATIONS ET TRANSITIONS

JQUERY ANIMATE

```
$('.btn-send').on('click', event => {
    $(event.target).prop('disabled', true);
    $('#animate').animate({
        height: 200,
        width: 400,
        opacity: 0.5
    }, 1000, "swing", () => $('.btn-send').trigger('sent'));
});
```

ANIMATIONS: CSS VS JS

https://css-tricks.com/myth-busting-css-animations-vs-javascript/

En ce qui concerne les performances, jQuery est beaucoup moins bien que du pure CSS ou d'autres librairies Javascript

Le CSS permet de faire tout un tas de transitions et d'animations, mais n'offre pas autant de possibilités que le Javascript (animer la scrollbar ?)

AVEZ VOUS DES QUESTIONS ?

MERCI

Openska