Construction d’applications web modernes

Faire face à l’évolution du web

Xavier Van de Woestyne

https://xvw.github.io

« J’aime bien la programmation »

OCaml, Haskell, **Elm**, **Erlang**, **Elixir**, **Ruby**, Io, **PHP**, **Go**!
Nim, Racket, **Ur/Web**, **Java**, F#

- **Elm** + **Elixir**
- API First
- Formation
- Bibliothèques Open Source

« useless software with useful languages »

- Meetup « bi-mensuel »
- Programmation fonctionnelle
- 25/05/2018 : Scala

J’adore Mike Brant, la bande-dessinée et la bière.

Objectifs

Histoire, Recherche, Méthodologies, Anticipation

Disclaimer

Limité à « **l’application Web** »,

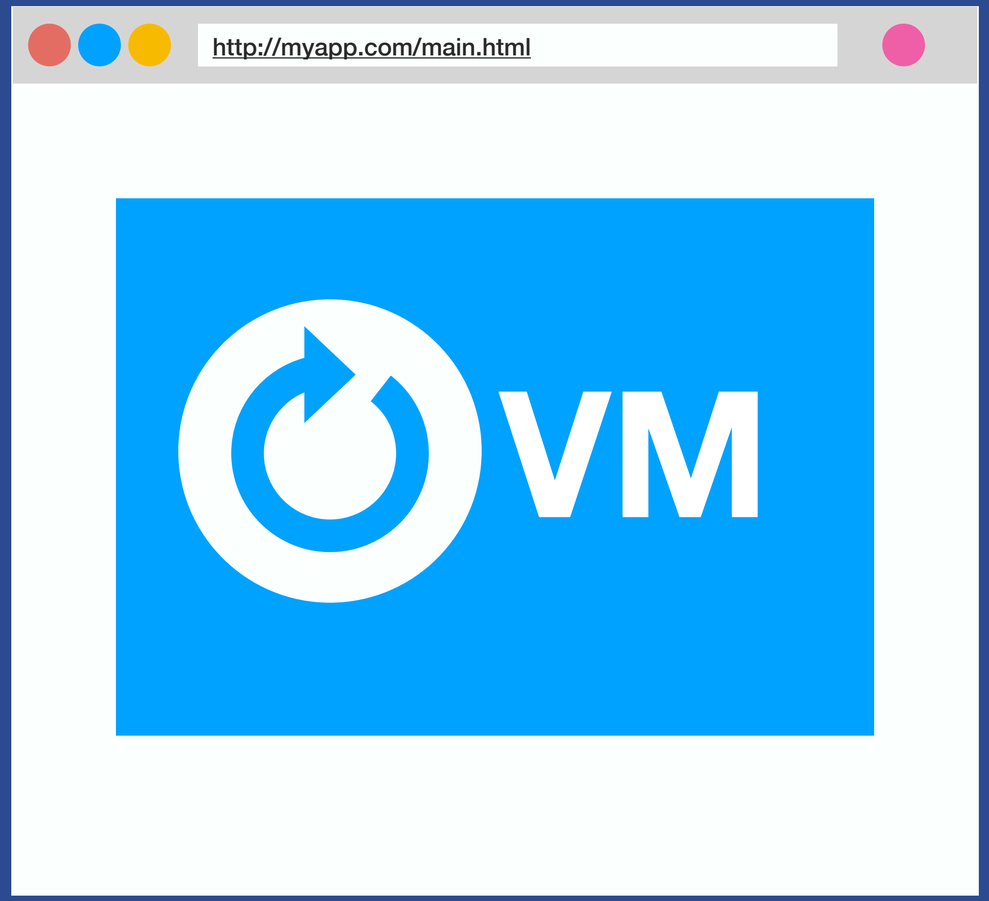
Sommaire

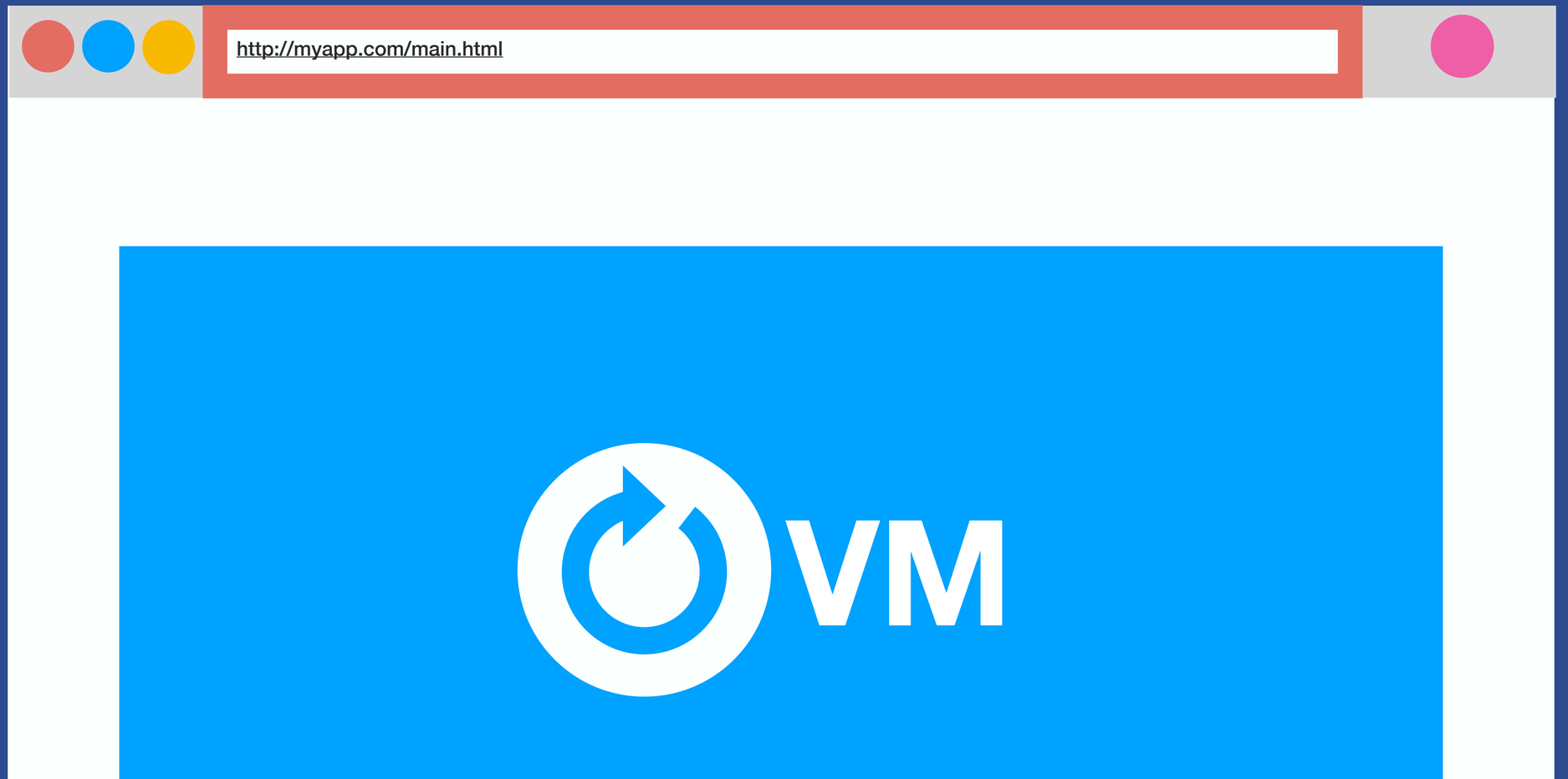
- Les premières application riches
- Entre applications natives et web
- Les caractéristiques d’une application web moderne
- Quelques framework avant-gardistes
- Mythes et réalités

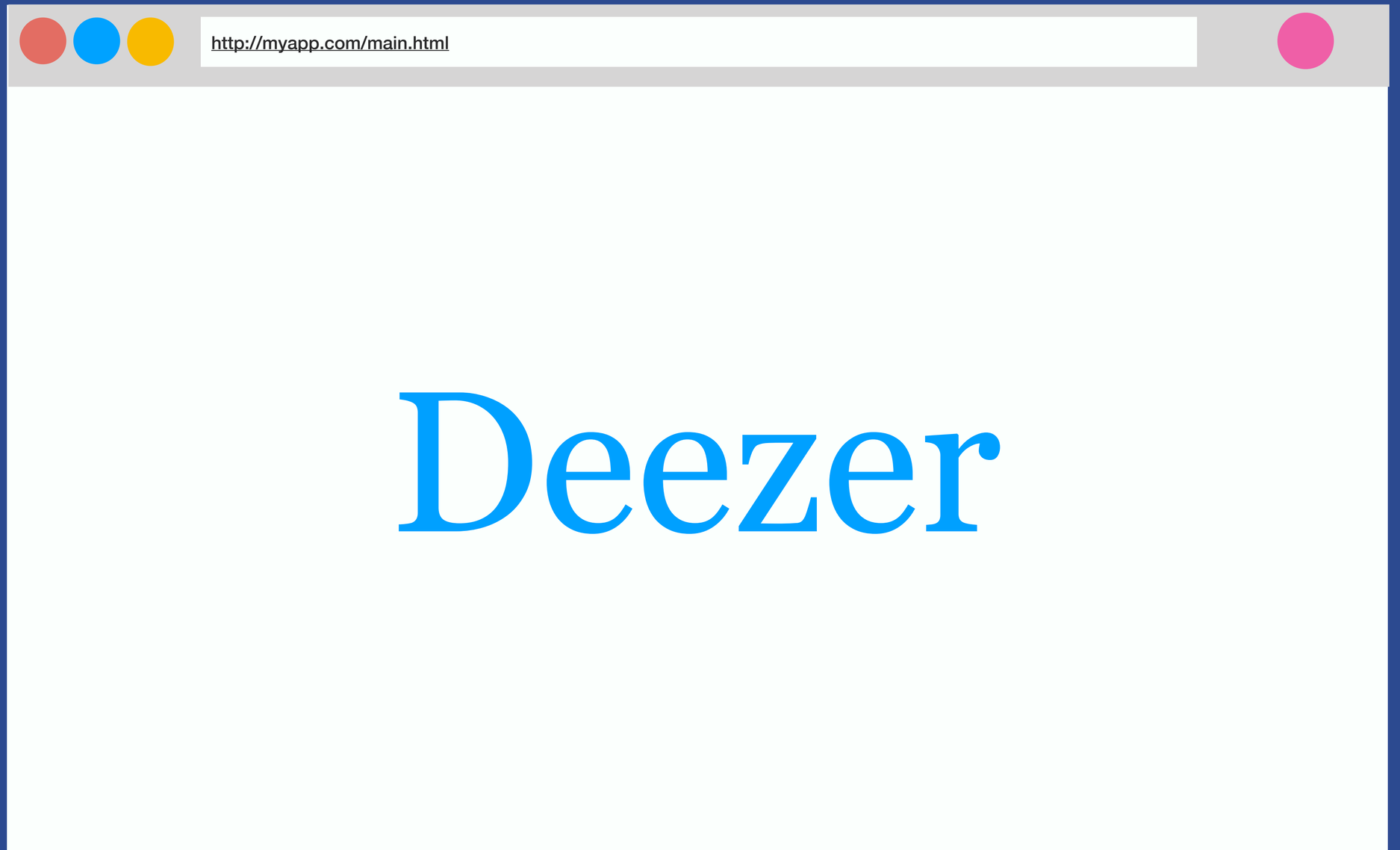
De 1989 à 2018
le web à évolué

Une frise grossière

Flash/Applets

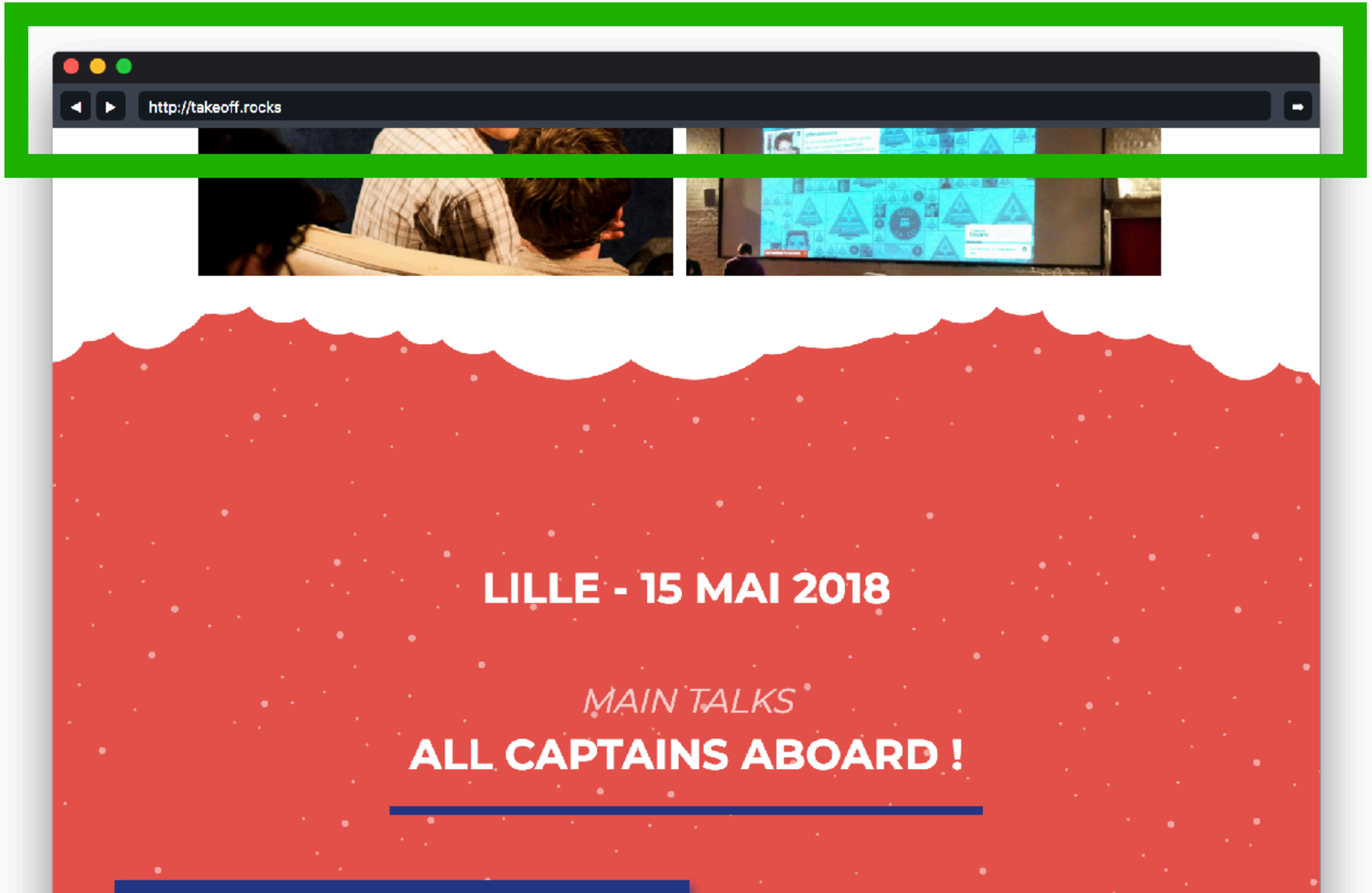



http://myapp.com/main.html
VM


http://myapp.com/main.html
Deezer

Ajax
CSS3
JavaScript+ES2015
SVG
HTML5
10000 frameworks
WebGL
WebSocket
Transpilation/compilation

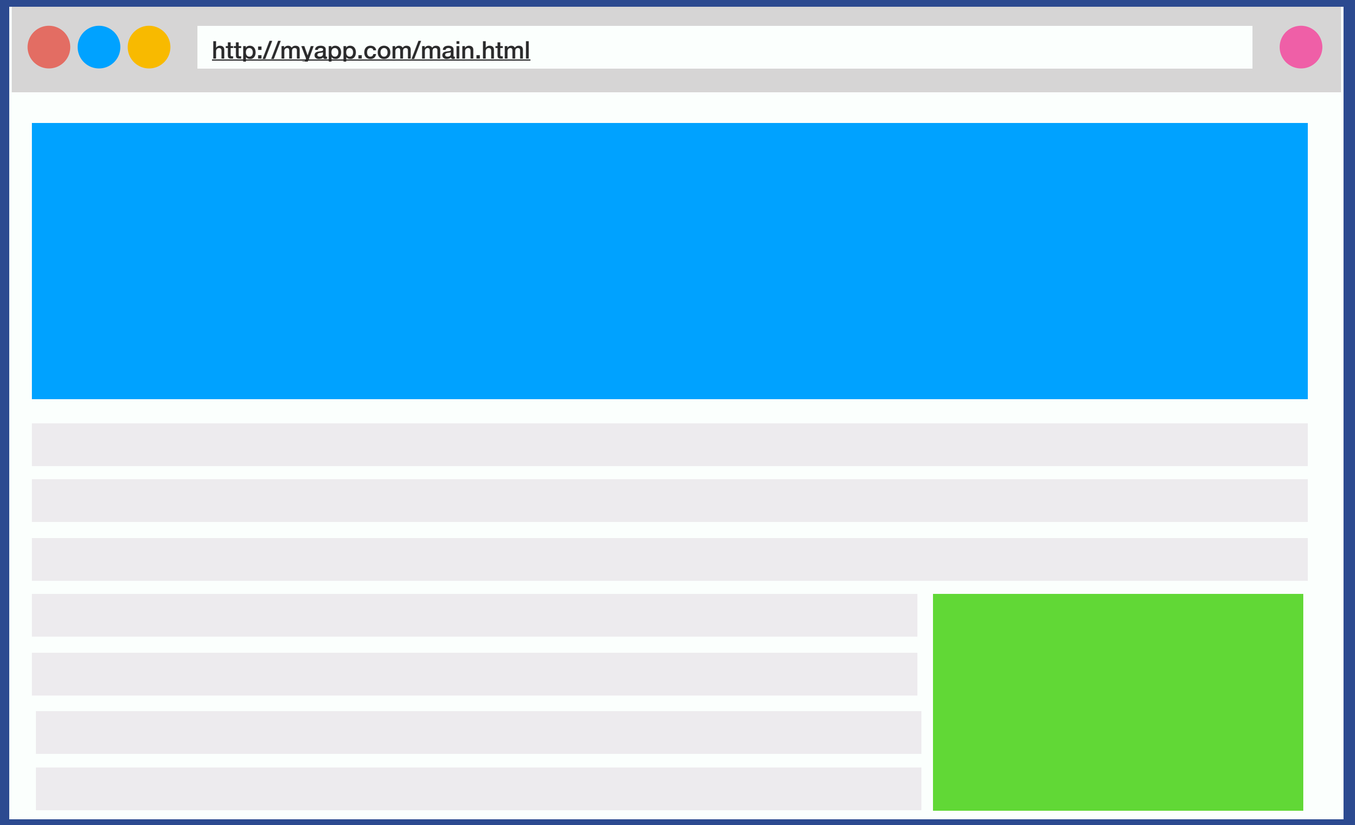
Une différence majeure



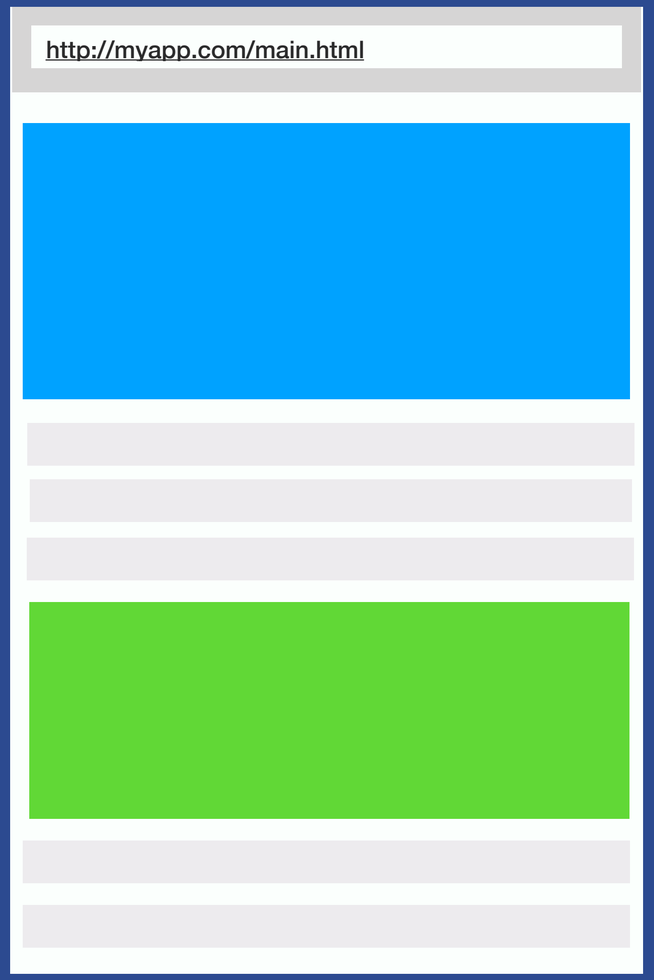
Les ingrédients

Multi-device
Accessible pour tous (et partout)
Temps réel souple
Tenant compte de son contexte
Réactive

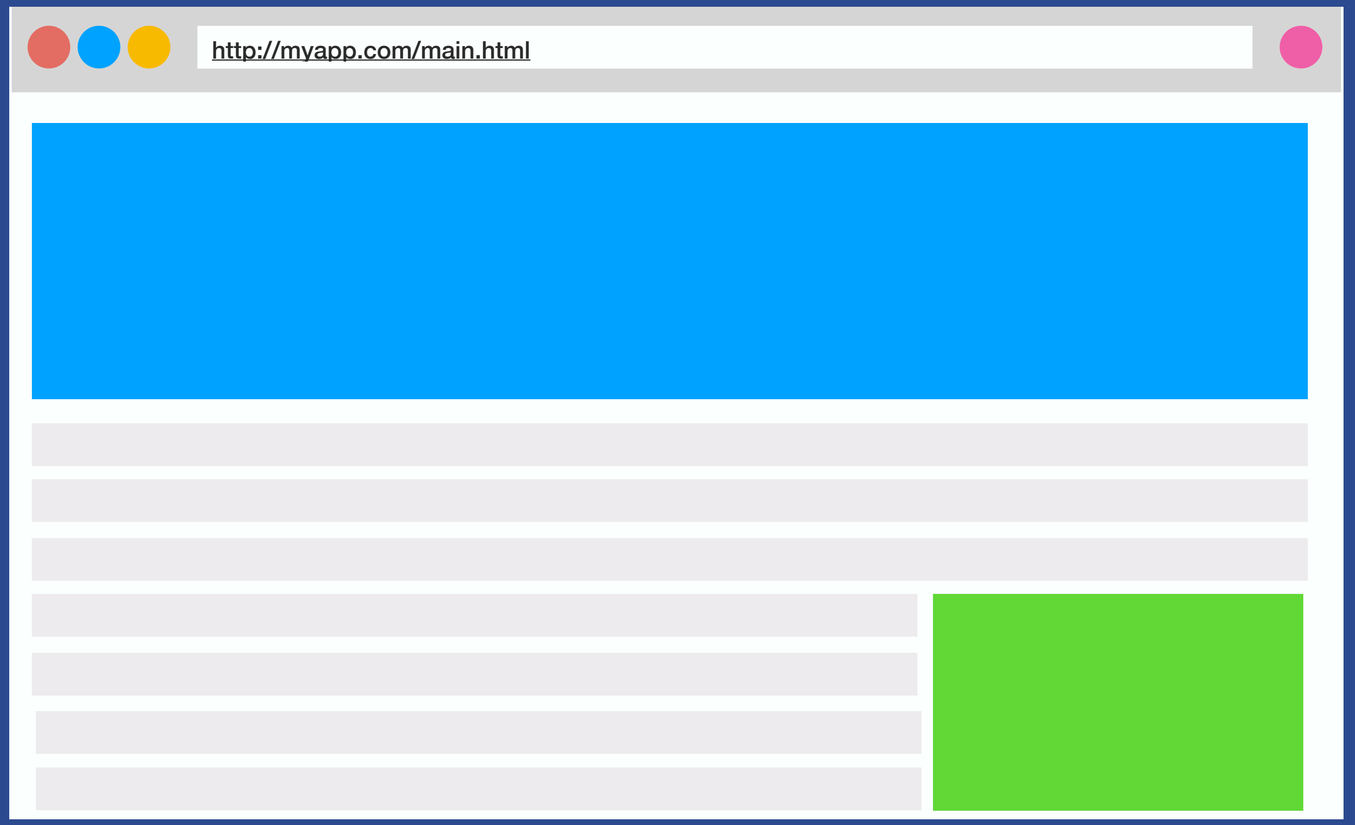
… et belle (mais ça c’est pas mon travail :v)



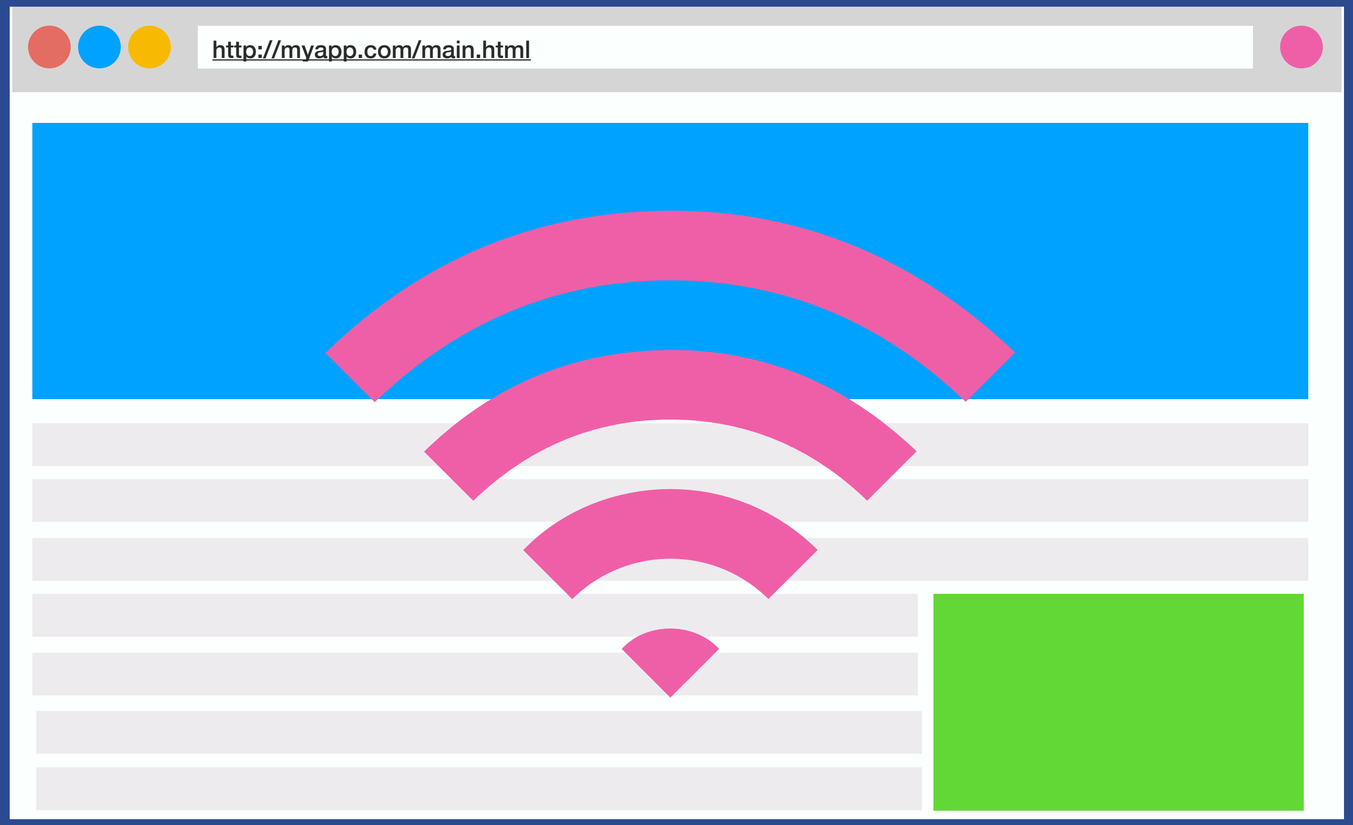
Multi-devices

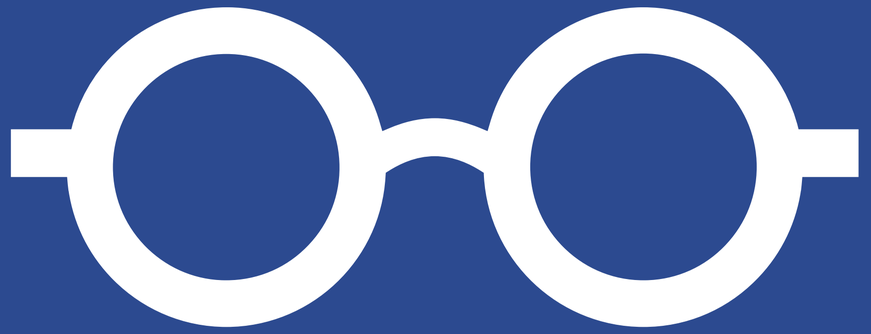


Multi-devices

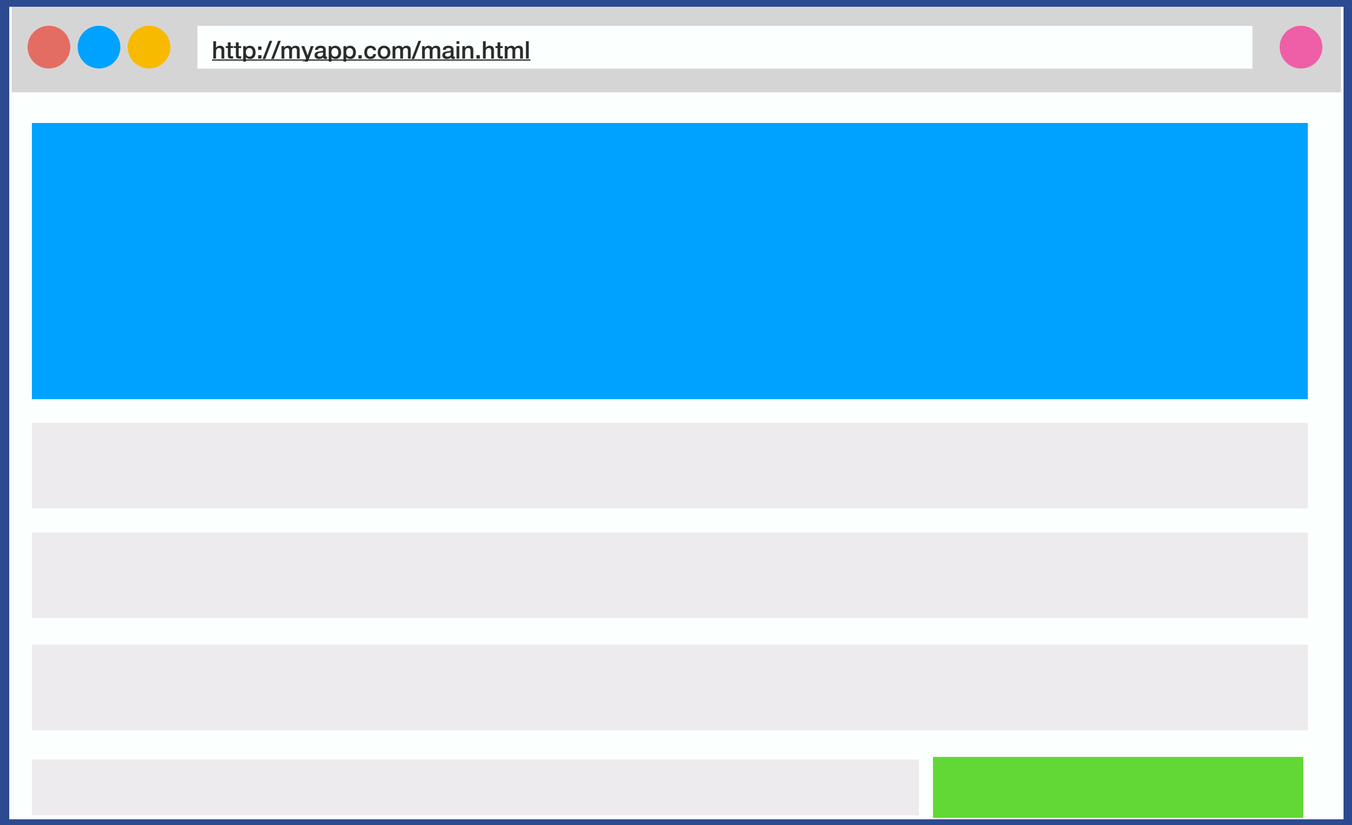


Multi-devices

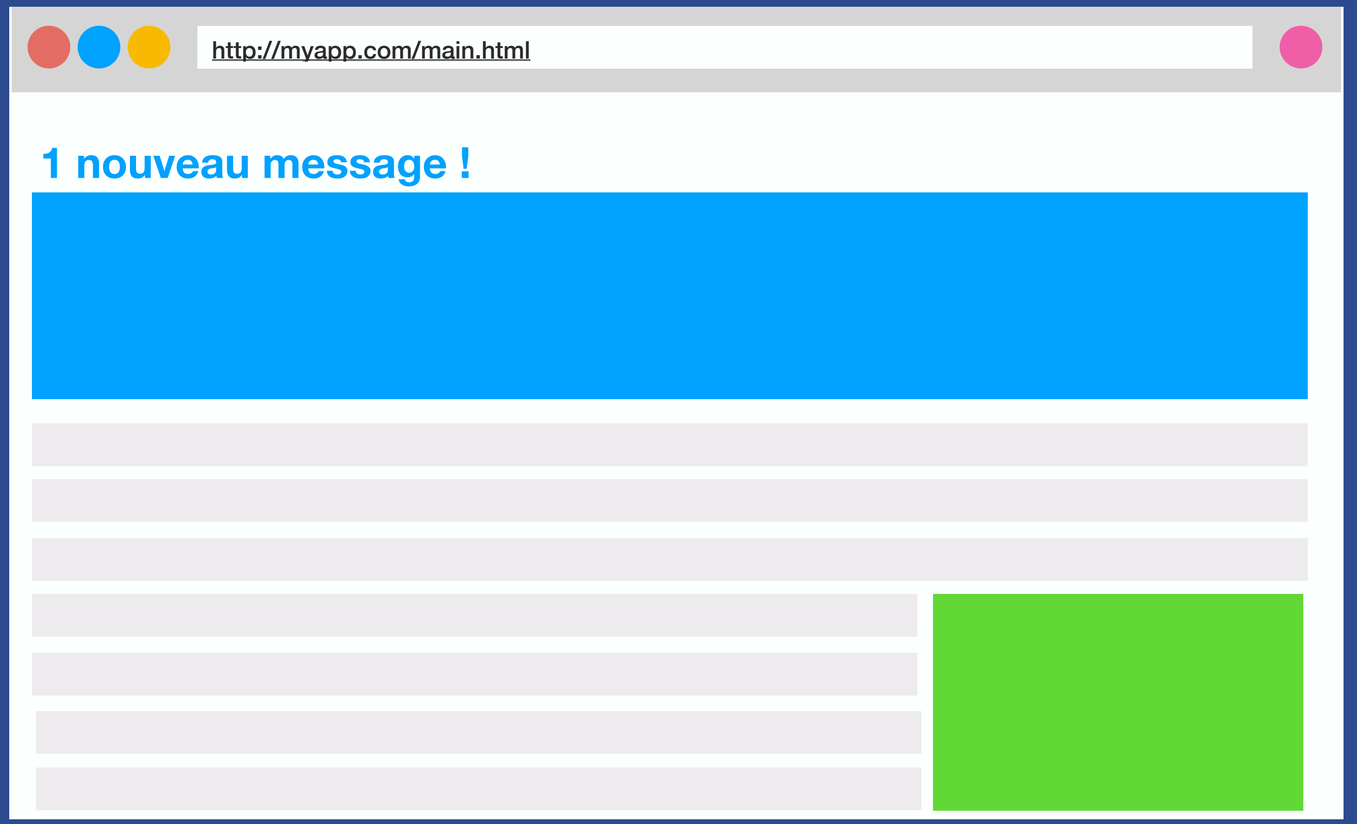




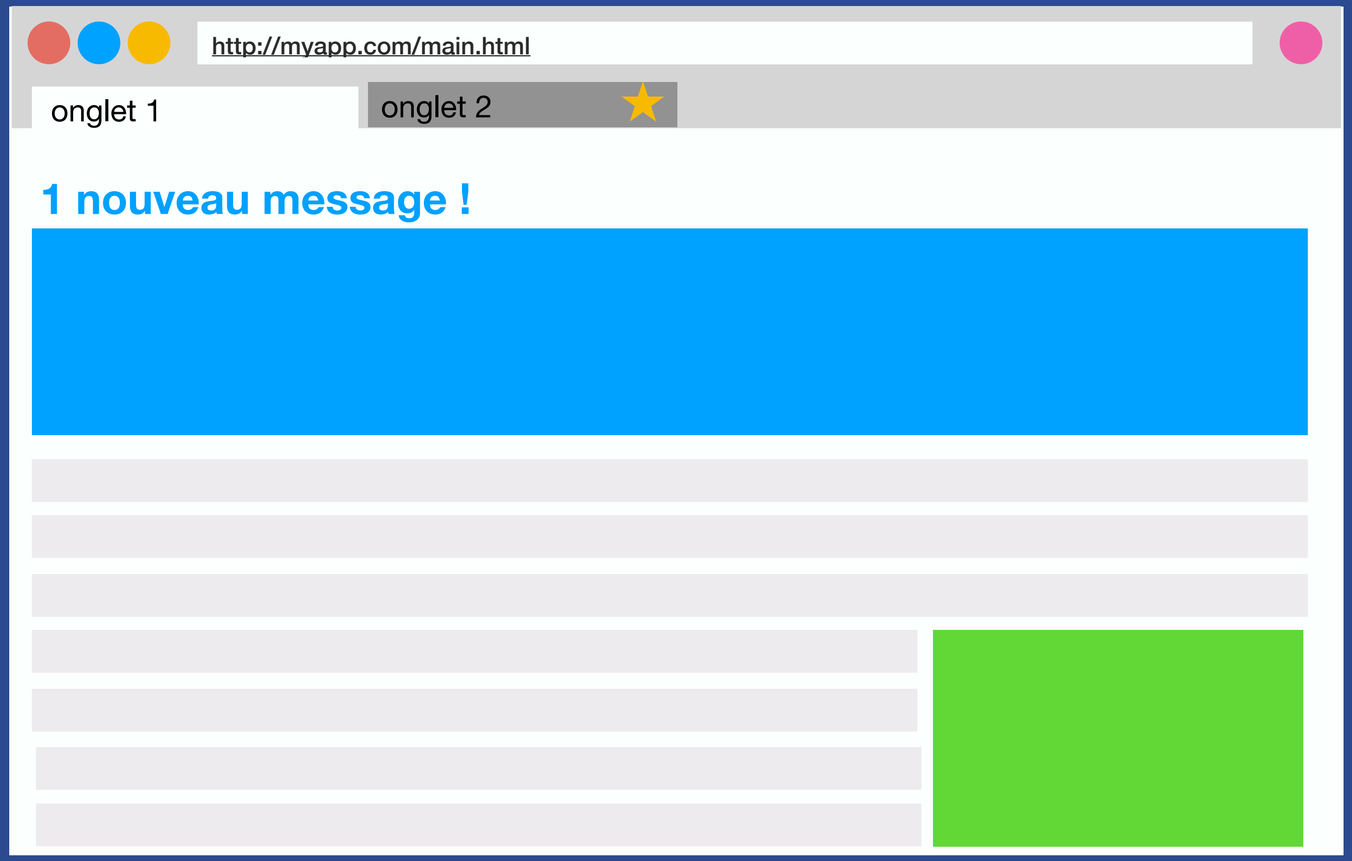
Accessible pour tous



Accessible pour tous



Temps réel



Contexte d’exécution et accessible partout

Profiter des accélérations possibles (SVG)

Utilisation d’un DOM Virtuel

Utilisation d’un DOM Incrémental

Utilisation de DirtyNodes

Une application réactive

Tierless VS API First


SUCH INNOVATION
MUCH WOW
memegenerator.net

1999 : HOP.js (framework/langage)

- Développé en Scheme
- Runtime compatible Scheme et NodeJS
- Ajout de « addEventListener » sur un objet « server »
- Intégration de noeud XML dans les vues (JSX)
- Tierless

2002 : Seaside (framework)

- SmallTalk (et Pharo)
- Tierless
- Initiation des continuations
- Initiation des « composants »

2004 : Ocsigen (framework/langage)

- OCaml
- Tierless (client, serveur, partagé)
- Réactif (Comet, Concurrence, OCaml-React)
- Extrêmement typé (html, requêtes, sql)
- Intégration des services (et co-services)

2007 : Lift (framework)

- Scala (lol)
- Tierless (client, serveur)
- Réactif (Comet)
- Presque bien typé

2009 : Links-lang (langage)

- OCaml
- Tierless (client, serveur, partagé)
- Réactif
- Extrêmement typé (html, requêtes, sql)
- « Essence of Web Abstraction »
- Intégration de Session Type

2014 : Ur/Web (langage)

- SML
- Tierless (client, serveur, partagé)
- Réactif
- Extrêmement typé (html, requêtes, sql)
- Très sécurisé

2015 : Elm (langage/framework)

- Haskell
- Only Front
- Réactif (FRP via Elm Architecture)

??? : PureScript (langage + framework)

- Haskell (en beau, genre pas paresseux)
- Only Front
- Effects via des monades (Aff et Eff)


WOW … RESEARCH !
« c’est comme ça que je t’aime »

Ici, je dois polémiquer sur le typage statique et sur les composants.

Fin. Merci <3 !