

JS

comme JavaScript

JS

HTML



CSS



JS



JS

JavaScript

- C'est un langage de programmation
- Ça n'a **RIEN** à voir avec JAVA 
- Ça a été créé en 10 jours en 1995 par un seul gars pour Netscape Navigator
- C'est depuis utilisé dans les navigateurs internet pour dynamiser le contenu des pages et interagir avec l'utilisateur.



JS

Principales utilisations dans le web

- De manière générale il sert à manipuler le HTML
- En **gros résumé**, c'est ce qui permet à la page d'interagir quand je clic quelque part ou que je saisie du texte
- Il permet de créer des applications web :
 - codepen.io 
 - [Youtube](https://www.youtube.com) 
 - [Gmail](https://mail.google.com) 
 - [Facebook](https://www.facebook.com) 
- De créer des Bannières HTML5 animées
- De placer des Cookies ou des pixels de tracking (GA etc ...)

JS

JavaScript: Hello World

- Un exemple :



- Cette popup est affichée par le code suivant :

```
alert('Coucou David');
```

JS

Comment je l'ajoute à ma page ?

- Soit à l'intérieur d'une balise `<script></script>` :

```
<script>
  // mon javascript
  var ma_variable = "toto"
</script>
```

- Soit directement dans un fichier séparé :

```
<script src="./js/mon_script.js"></script>
```

JS

Rappel : Page HTML Minimale



index.html ✘

```
1  <!DOCTYPE html>
2  ⚡ <html lang="en">
3  ⚡   ⚡ <head>
4  ⚡     ⚡   <title>Titre de la page</title>
5  ⚡     ⚡   <meta charset="UTF-8">
6  ⚡     ⚡   <script src="js/script.js"></script>
7  ⚡     ⚡   <link href="css/style.css" rel="stylesheet">
8  ⚡   ⚡ </head>
9  ⚡   ⚡ <body>
10 ⚡     ⚡   Mon contenu !
11 ⚡     ⚡ </body>
12 ⚡   ⚡ </html>
13 ⚡
```



Aparté

- Sinon ... bah ... en fait c'est pas si dur que ça n'y paraît, c'est comme apprendre une nouvelle langue, sauf que là on connaît déjà les mots ...



JS

Les bases du language

- Vous trouverez une documentation très complète à cette adresse :

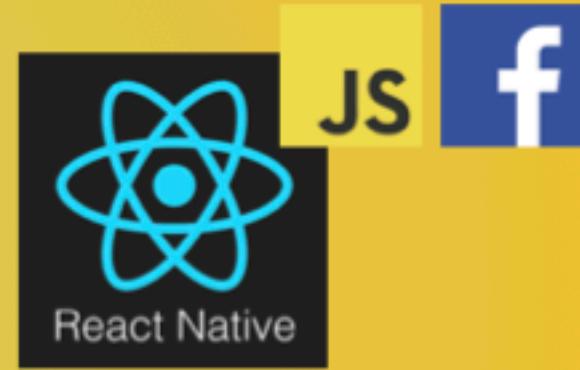


<https://developer.mozilla.org/fr/docs/Web/JavaScript>

JS

JavaScript: Popularité

- En développement mobile :



JavaScript: Popularité

- En développement mobile :



JS

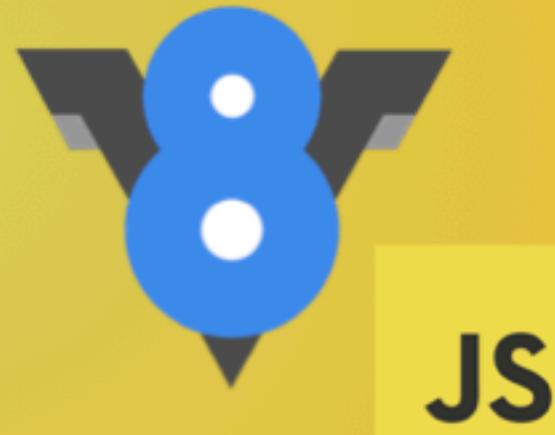


JS

JS

JavaScript: Popularité

- En développement d'application bureau et serveur :

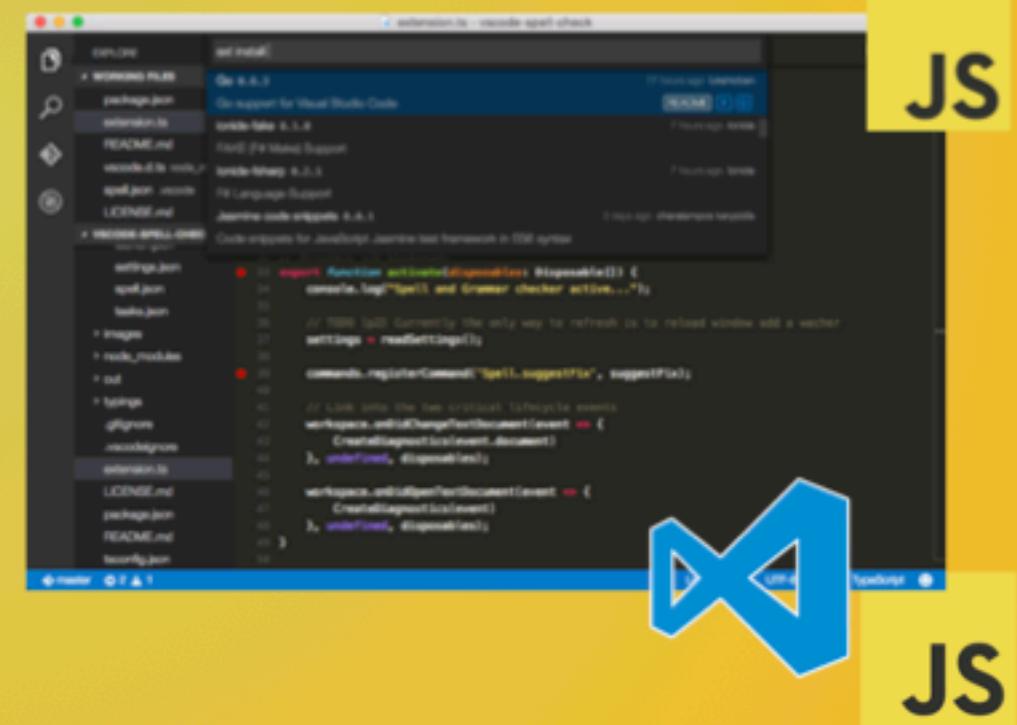


JavaScript: Popularité

- En développement d'application bureau et serveur :



JavaScript: Popularité



JS

JS

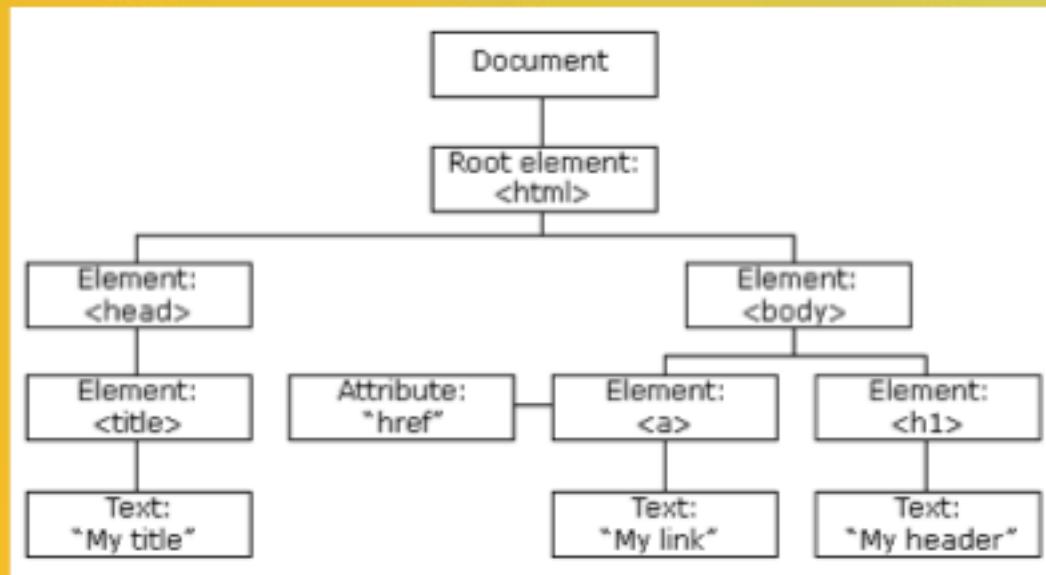
JS

JavaScript dans le web

JS

JavaScript dans le web

- A quoi sert JavaScript dans les pages web ?
- Il sert à interagir avec le DOM (Document Object Model)
 - Le dom, en gros c'est l'arbre du HTML d'une page

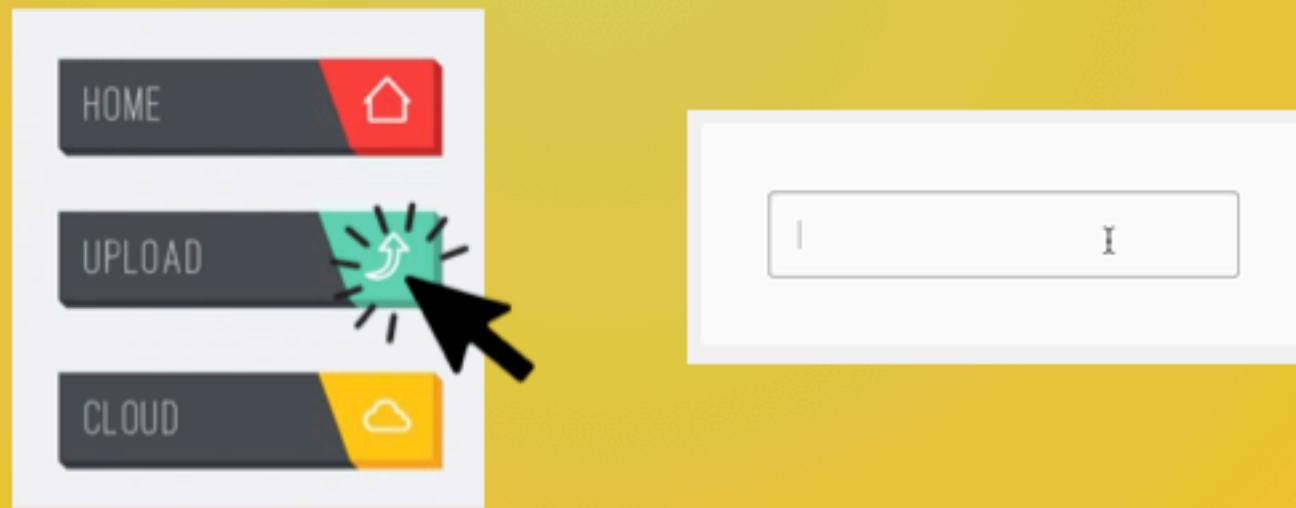


```
index.html x
1  <!DOCTYPE html>
2  ⋮ <html lang="en">
3  ⋮   ⋮ <head>
4  ⋮     ⋮   <title>Titre de la page</title>
5  ⋮     ⋮   <meta charset="UTF-8">
6  ⋮     ⋮   <script src="js/script.js"></script>
7  ⋮     ⋮   <link href="css/style.css" rel="stylesheet">
8  ⋮   ⋮ </head>
9  ⋮ ⋮ <body>
10 ⋮   ⋮   Mon contenu !
11 ⋮   ⋮ </body>
12 ⋮ </html>
13
```

The code shows the structure of the 'index.html' file. It defines an HTML document with a title 'Titre de la page', a meta charset 'UTF-8', a script from 'js/script.js', and a stylesheet from 'css/style.css'. The body of the page contains the text 'Mon contenu !'.

JavaScript dans le web

- Si vous n'avez rien compris c'est pas très grave.
- Ce qu'il faut retenir c'est qu'avec JavaScript je vais pouvoir lire et écrire le contenu de ma page HTML et écouter des événements comme les clicks ou la saisie au clavier.



HTML



<input>

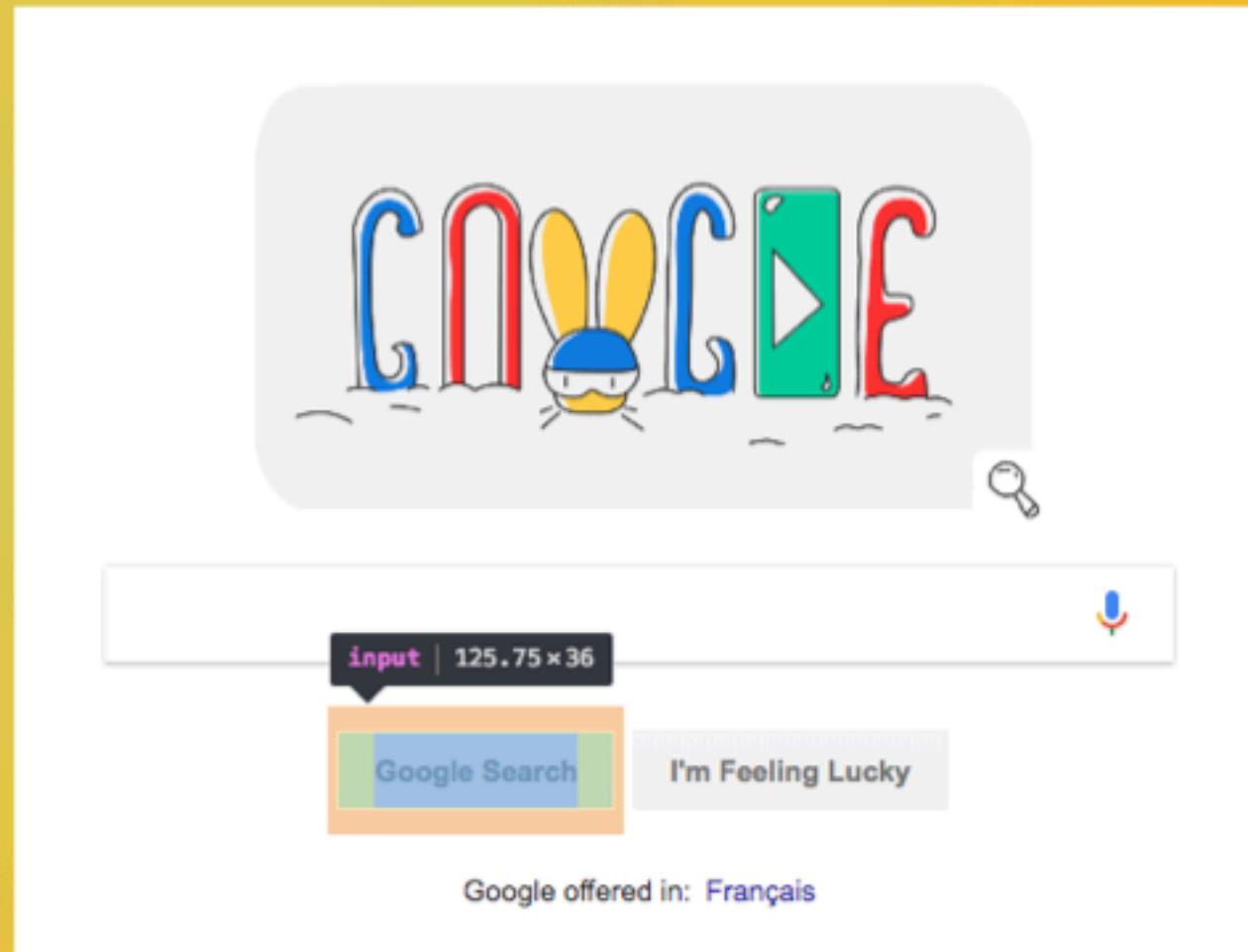
Sélectionner un élément

<button>

JS

Sélectionner un élément

- Je veux « attraper » un élément de ma page
- ici par exemple je veux récupérer ce bouton pour y accéder dans mon JavaScript



JavaScript dans le web

- Une **seule** méthode magique permet de sélectionner un élément du HTML pour l'utiliser en JavaScript :



```
document.querySelector("mon_element");
```

- On va la décortiquer
- ... mais ne l'apprenez pas encore par coeur ...

JS

JavaScript dans le web

```
document.querySelector("mon_element");
```

C'est **l'objet** qui
représente la page Web

On invoque une fonction sur cet objet
(qu'on appelle une méthode)

On lui passe en argument
l'élément que l'on recherche



JavaScript dans le web

- Sauf que l'élément qu'on recherche ... "mon_element"

- Est un sélecteur CSS



```
document.querySelector("#toto");
document.querySelector(".green");
document.querySelector("h1");
document.querySelector("#toto > h1 + a");
```



Sélectionner un élément

```
<div id="toto">
  <div class="green">blablabla</div>
  <h1>Titre</h1>
  <a href="http://google.com">aller sur google</a>
</div>

var element1 = document.querySelector("#toto");
var element2 = document.querySelector(".green");
var element3 = document.querySelector("h1");
var element4 = document.querySelector("#toto > h1 + a");
```



la valeur de retour est un **objet** qui représente une balise

JS

Petite fonction raccourcie

- On est d'accord que c'est un peu chiant à écrire ce truc
- Et si on le mettait dans une fonction ?

```
function $ ( element_cherché ) {  
    var element_trouvé = document.querySelector( element_cherché );  
    return element_trouvé;  
}
```

- On aurait pu l'appeler comme on veut, mais on va l'appeler « **dollar** » et utiliser le symbol « **\$** » pour l'appeler

JS

Petite fonction raccourcie

```
function $( element_cherché ) {  
    var element_trouvé = document.querySelector( element_cherché );  
    return element_trouvé;  
}
```



```
function $( element_cherché ) {  
    return document.querySelector( element_cherché );  
}
```

- C'est exactement pareil, mais plus facile à écrire
- **Note:** Ça n'a **rien** à voir avec le **\$** de PHP, j'aurais pu l'appelée rachid

JS

Sélectionner un élément

```
function $( element_cherché ) {  
    return document.querySelector( element_cherché );  
}
```

```
<div id="toto">  
    <div class="green">blablabla</div>  
    <h1>Titre</h1>  
    <a href="http://google.com">aller sur google</a>  
</div>
```

```
var element1 = $("#toto");  
var element2 = $(".green");  
var element3 = $("h1");  
var element4 = $("#toto > h1 + a");
```

JS

Resumé

- On sélectionne un élément avec la fonction « `$()` » :

```
var element1 = $("#toto");
var element2 = $(".green");
var element3 = $("h1");
var element4 = $("#toto > h1 + a");
```

- Cette fonction prend un paramètre « chaîne » qui correspond au **sélecteur css** de l'élément qu'on souhaite récupérer.
- On récupère cet objet qu'on peut mettre dans une **variable**

JS

. green

Changer les classes dynamiquement

. yellow

. red

JS

Resumé

- On sélectionne un élément avec la fonction « `$()` » :

```
var element1 = $("#toto");
var element2 = $(".green");
var element3 = $("h1");
var element4 = $("#toto > h1 + a");
```

- Cette fonction prend un paramètre « chaîne » qui correspond au **sélecteur css** de l'élément qu'on souhaite récupérer.
- On récupère cet objet qu'on peut mettre dans une **variable**

JS

Changer les classes dynamiquement

- En JavaScript, on a vu qu'il est possible d'éditer dynamiquement des attributs de balises avec la méthode « **setAttribute()** »
- En HTML on à vu que l'attribut « **class** » qui permet d'assigner des classes CSS, pouvais contenir plusieurs éléments :

```
<span class="souligne barre rouge gras">Mon texte</span>
```

- Sauf que si je fais un **setAttribute** sur **class**, j'écrase toutes les autres classes :

```
$( 'span' ).setAttribute('class', 'jaune');
```

```
<span class="jaune">Mon texte</span>
```

JS

Resumé

- On sélectionne un élément avec la fonction « `$()` » :

```
var element1 = $("#toto");
var element2 = $(".green");
var element3 = $("h1");
var element4 = $("#toto > h1 + a");
```

- Cette fonction prend un paramètre « chaîne » qui correspond au **sélecteur css** de l'élément qu'on souhaite récupérer.
- On récupère cet objet qu'on peut mettre dans une **variable**

JS

Changer les classes dynamiquement

- À la place on va utiliser l'élément « **classList** » qui possède plusieurs méthodes :

balise HTML

classList

add(class)

remove(class)

```
$('span').classList.add( 'jaune' );
```

```
$('span').classList.remove( 'rouge' );
```

```
<span class="souligne barre gras jaune">Mon texte</span>
```

JS

Changer les classes dynamiquement

```
<span class="souligne barre rouge gras">Mon texte</span>
```

- La dernière classe ajoutée se mettent à la fin, mais ça n'a aucune incidence sur les autres classes appliquées

```
$( 'span' ).classList.add( 'jaune' );
```

```
$( 'span' ).classList.remove( 'rouge' );
```

```
<span class="souligne barre gras jaune">Mon texte</span>
```

JS

Resumé

- On sélectionne un élément avec la fonction « `$()` » :

```
var element1 = $("#toto");
var element2 = $(".green");
var element3 = $("h1");
var element4 = $("#toto > h1 + a");
```

- Cette fonction prend un paramètre « chaîne » qui correspond au **sélecteur css** de l'élément qu'on souhaite récupérer.
- On récupère cet objet qu'on peut mettre dans une **variable**

JS

Changer les classes dynamiquement

- Il est également possible d'utiliser la méthode **toggle** sur « **classList** », si la classe est présente on l'enlève, sinon on l'ajoute :

```
<span class="souligne barre gras jaune">Mon texte</span>

$('span').classList.toggle( 'gras' );

<span class="souligne barre jaune">Mon texte</span>

$('span').classList.toggle( 'gras' );

<span class="souligne barre jaune gras">Mon texte</span>
```

JS

Résumé

balise HTML

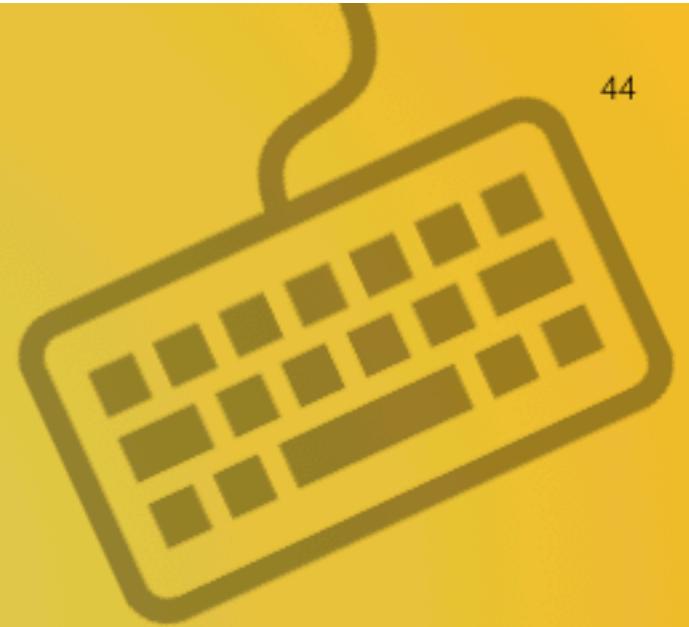
classList

`add(class)`

`remove(class)`

`toggle(class)`

- Il est possible de manipuler les classes appliquées à une balise HTML avec l'objet classList qui possède 3 méthodes :
 - add** : permet d'ajouter une classe à l'élément
 - remove** : permet d'enlever une classe à l'élément
 - toggle** : permet de toggle une classe



Travaux pratiques

JS

Hé beh non, pas tout de suite ...

JS



Écouter des événements

JS



Écouter des événements

- Il est possible d'écouter des événements qui arrivent à un élément et d'exécuter une fonction lorsque cet événement se produit. On utiliser la méthode suivante **sur** notre objet :

```
$( '#mon_element' ).addEventListener( 'evenement', nom_fonction );
```

- L'« **evenement** » peut être de plusieurs types, mais les principaux sont « **click** » et « **input** » (saisie de texte).

JS



Écouter des événements

```
$('mon_element').addEventListener('evenement', nom_fonction);
```



C'est **l'objet** qui représente
notre élément HTML
(C'est une balise de notre code HTML)



On invoque une méthode pour écouter
des événements sur cet **objet**.
Cette méthode prend 2 paramètres

JS



Écouter des événements

```
$( '#mon_element' ).addEventListener( 'evenement', nom_fonction );
```



balise HTML



addEventListerner(event, function)

JS

Écouter des événements

```
$('mon_element').addEventListener('evenement', nom_fonction );
```



On lui passe en premier argument l'événement que l'on souhaite surveiller ("click", "input", ...)

Et en deuxième argument le nom d'une **fonction** que l'on veut exécuter lors de l'événement.

Elle ne prend **pas** de paramètres ou de parenthèses

JS



Écouter des événements

```
$( '#mon_element' ).addEventListener( 'evenement', nom_fonction );
```

- Exemple concret :

```
$("#bouton").addEventListener('click', afficher );

function afficher() {
    console.log( 'afficher appelée' );
}
```

JS

Petite fonction raccourcie

```
$(element).addEventListener('event', fonction );
```

- On va se faire un alias de ça aussi :

```
Object.prototype.on = function (e,cb) {  
    this.addEventListener(e,cb);  
}
```

- Cela nous permet d'écrire ce code, qui fait la même chose :

```
$(element).on('event', fonction );
```

JS

Petite fonction raccourcie

- Si je reprend l'exemple précédent :

```
$("#bouton").addEventListener('click', afficher );  
  
function afficher() {  
    console.log( 'afficher appelée' );  
}
```

```
$("#bouton").on('click', afficher );  
  
function afficher() {  
    console.log( 'afficher appelée' );  
}
```



JS

Petite fonction raccourcie

- Les alias c'est sympa, grâce à eux, ces deux lignes sont exactement identique dans nos TPs. Cool non ?

```
document.querySelector("#id_elem").addEventListener('click', ma_fonction );  
$("#id_elem").on('click', ma_fonction );
```

JS

.green

Changer les classes dynamiquement

.yellow

.red

JS

Changer les classes dynamiquement

- En JavaScript, on a vu qu'il est possible d'éditer dynamiquement des attributs de balises avec la méthode « **setAttribute()** »
- En HTML on à vu que l'attribut « **class** » qui permet d'assigner des classes CSS, pouvais contenir plusieurs éléments :

```
<span class="souligne barre rouge gras">Mon texte</span>
```

- Sauf que si je fais un **setAttribute** sur **class**, j'écrase toutes les autres classes :

```
$( 'span' ).setAttribute('class', 'jaune');
```

```
<span class="jaune">Mon texte</span>
```

JS

Changer les classes dynamiquement

- À la place on va utiliser l'élément « **classList** » qui possède plusieurs méthodes :

balise HTML

classList

add(class)

remove(class)

```
$('span').classList.add( 'jaune' );
```

```
$('span').classList.remove( 'rouge' );
```

```
<span class="souligne barre gras jaune">Mon texte</span>
```

JS

Changer les classes dynamiquement

```
<span class="souligne barre rouge gras">Mon texte</span>
```

- La dernière classe ajoutée se mettent à la fin, mais ça n'a aucune incidence sur les autres classes appliquées

```
$( 'span' ).classList.add( 'jaune' );
```

```
$( 'span' ).classList.remove( 'rouge' );
```

```
<span class="souligne barre gras jaune">Mon texte</span>
```

JS

Changer les classes dynamiquement

- Il est également possible d'utiliser la méthode **toggle** sur « **classList** », si la classe est présente on l'enlève, sinon on l'ajoute :

```
<span class="souligne barre gras jaune">Mon texte</span>

$('span').classList.toggle( 'gras' );

<span class="souligne barre jaune">Mon texte</span>

$('span').classList.toggle( 'gras' );

<span class="souligne barre jaune gras">Mon texte</span>
```

JS

Résumé

balise HTML

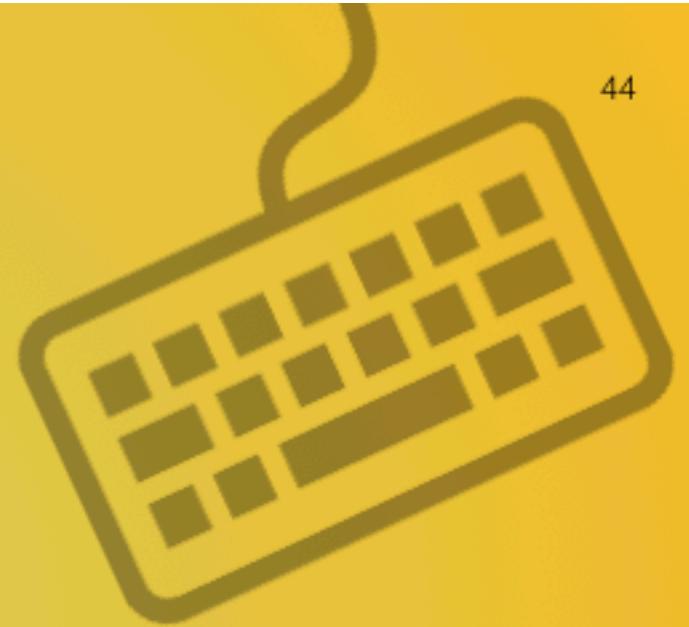
classList

`add(class)`

`remove(class)`

`toggle(class)`

- Il est possible de manipuler les classes appliquées à une balise HTML avec l'objet classList qui possède 3 méthodes :
 - add** : permet d'ajouter une classe à l'élément
 - remove** : permet d'enlever une classe à l'élément
 - toggle** : permet de toggle une classe



Travaux pratiques

JS

Hé beh non, pas tout de suite ...

JS



Écouter des événements

JS



Écouter des événements

- Il est possible d'écouter des événements qui arrivent à un élément et d'exécuter une fonction lorsque cet événement se produit. On utiliser la méthode suivante **sur** notre objet :

```
$( '#mon_element' ).addEventListener( 'evenement', nom_fonction );
```

- L'« **evenement** » peut être de plusieurs types, mais les principaux sont « **click** » et « **input** » (saisie de texte).

JS



Écouter des événements

```
$('mon_element').addEventListener('evenement', nom_fonction);
```



C'est **l'objet** qui représente
notre élément HTML
(C'est une balise de notre code HTML)



On invoque une méthode pour écouter
des événements sur cet **objet**.
Cette méthode prend 2 paramètres

JS



Écouter des événements

```
$( '#mon_element' ).addEventListener( 'evenement', nom_fonction );
```



balise HTML



addEventListerner(event, function)

JS

Écouter des événements

```
$('mon_element').addEventListener('evenement', nom_fonction );
```



On lui passe en premier argument l'événement que l'on souhaite surveiller ("click", "input", ...)

Et en deuxième argument le nom d'une **fonction** que l'on veut exécuter lors de l'événement.

Elle ne prend **pas** de paramètres ou de parenthèses

JS



Écouter des événements

```
$( '#mon_element' ).addEventListener( 'evenement', nom_fonction );
```

- Exemple concret :

```
$("#bouton").addEventListener('click', afficher );

function afficher() {
    console.log( 'afficher appelée' );
}
```

JS

Petite fonction raccourcie

```
$(element).addEventListener('event', fonction );
```

- On va se faire un alias de ça aussi :

```
Object.prototype.on = function (e,cb) {  
    this.addEventListener(e,cb);  
}
```

- Cela nous permet d'écrire ce code, qui fait la même chose :

```
$(element).on('event', fonction );
```

JS

Petite fonction raccourcie

- Si je reprend l'exemple précédent :

```
$("#bouton").addEventListener('click', afficher );  
  
function afficher() {  
    console.log( 'afficher appelée' );  
}
```

```
$("#bouton").on('click', afficher );  
  
function afficher() {  
    console.log( 'afficher appelée' );  
}
```



JS

Petite fonction raccourcie

- Les alias c'est sympa, grâce à eux, ces deux lignes sont exactement identique dans nos TPs. Cool non ?

```
document.querySelector("#id_elem").addEventListener('click', ma_fonction );  
$("#id_elem").on('click', ma_fonction );
```

JS

. green

Changer les classes dynamiquement

. yellow

. red

JS

Changer les classes dynamiquement

- En JavaScript, on a vu qu'il est possible d'éditer dynamiquement des attributs de balises avec la méthode « **setAttribute()** »
- En HTML on à vu que l'attribut « **class** » qui permet d'assigner des classes CSS, pouvais contenir plusieurs éléments :

```
<span class="souligne barre rouge gras">Mon texte</span>
```

- Sauf que si je fais un **setAttribute** sur **class**, j'écrase toutes les autres classes :

```
$( 'span' ).setAttribute('class', 'jaune');
```

```
<span class="jaune">Mon texte</span>
```

JS

Changer les classes dynamiquement

- À la place on va utiliser l'élément « **classList** » qui possède plusieurs méthodes :

balise HTML

classList

add(class)

remove(class)

```
$('span').classList.add( 'jaune' );
```

```
$('span').classList.remove( 'rouge' );
```

```
<span class="souligne barre gras jaune">Mon texte</span>
```

JS

Changer les classes dynamiquement

```
<span class="souligne barre rouge gras">Mon texte</span>
```

- La dernière classe ajoutée se mettent à la fin, mais ça n'a aucune incidence sur les autres classes appliquées

```
$( 'span' ).classList.add( 'jaune' );
```

```
$( 'span' ).classList.remove( 'rouge' );
```

```
<span class="souligne barre gras jaune">Mon texte</span>
```

JS

Changer les classes dynamiquement

- Il est également possible d'utiliser la méthode **toggle** sur « **classList** », si la classe est présente on l'enlève, sinon on l'ajoute :

```
<span class="souligne barre gras jaune">Mon texte</span>

$('span').classList.toggle( 'gras' );

<span class="souligne barre jaune">Mon texte</span>

$('span').classList.toggle( 'gras' );

<span class="souligne barre jaune gras">Mon texte</span>
```

JS

Résumé

balise HTML

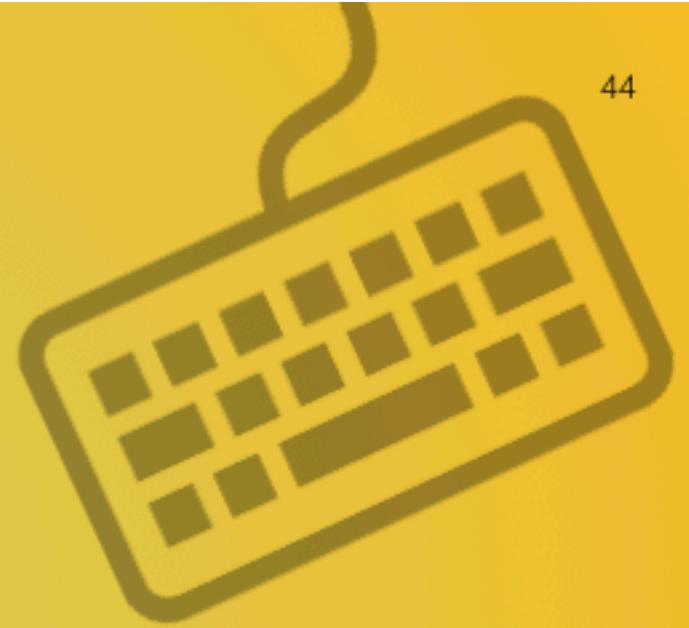
classList

`add(class)`

`remove(class)`

`toggle(class)`

- Il est possible de manipuler les classes appliquées à une balise HTML avec l'objet classList qui possède 3 méthodes :
 - add** : permet d'ajouter une classe à l'élément
 - remove** : permet d'enlever une classe à l'élément
 - toggle** : permet de toggle une classe



Travaux pratiques

JS

Hé beh non, pas tout de suite ...

JS



Écouter des événements

JS



Écouter des événements

- Il est possible d'écouter des événements qui arrivent à un élément et d'exécuter une fonction lorsque cet événement se produit. On utiliser la méthode suivante **sur** notre objet :

```
$( '#mon_element' ).addEventListener( 'evenement', nom_fonction );
```

- L'« **evenement** » peut être de plusieurs types, mais les principaux sont « **click** » et « **input** » (saisie de texte).

JS



Écouter des événements

```
$('mon_element').addEventListener('evenement', nom_fonction);
```



C'est **l'objet** qui représente
notre élément HTML
(C'est une balise de notre code HTML)



On invoque une méthode pour écouter
des événements sur cet **objet**.
Cette méthode prend 2 paramètres

JS



Écouter des événements

```
$( '#mon_element' ).addEventListener( 'evenement', nom_fonction );
```



balise HTML



addEventListerner(event, function)

JS

Écouter des événements

```
$('mon_element').addEventListener('evenement', nom_fonction );
```



On lui passe en premier argument
l'événement que l'on souhaite
surveiller ("click", "input", ...)

Et en deuxième argument le nom
d'une **fonction** que l'on veut exécuter
lors de l'événement.

Elle ne prend **pas** de paramètres
ou de parenthèses

JS



Écouter des événements

```
$( '#mon_element' ).addEventListener( 'evenement', nom_fonction );
```

- Exemple concret :

```
$("#bouton").addEventListener('click', afficher );

function afficher() {
    console.log( 'afficher appelée' );
}
```

JS

Petite fonction raccourcie

```
$(element).addEventListener('event', fonction );
```

- On va se faire un alias de ça aussi :

```
Object.prototype.on = function (e,cb) {  
    this.addEventListener(e,cb);  
}
```

- Cela nous permet d'écrire ce code, qui fait la même chose :

```
$(element).on('event', fonction );
```

JS

Petite fonction raccourcie

- Si je reprend l'exemple précédent :

```
$("#bouton").addEventListener('click', afficher );  
  
function afficher() {  
    console.log( 'afficher appelée');  
}
```

```
$("#bouton").on('click', afficher );  
  
function afficher() {  
    console.log( 'afficher appelée');  
}
```



JS

Petite fonction raccourcie

- Les alias c'est sympa, grâce à eux, ces deux lignes sont exactement identique dans nos TPs. Cool non ?

```
document.querySelector("#id_elem").addEventListener('click', ma_fonction );  
$("#id_elem").on('click', ma_fonction );
```

JS

Resumé

- On sélectionne un élément avec la fonction « `$()` » :

```
var element1 = $("#toto");
var element2 = $(".green");
var element3 = $("h1");
var element4 = $("#toto > h1 + a");
```

- Cette fonction prend un paramètre « chaîne » qui correspond au **sélecteur css** de l'élément qu'on souhaite récupérer.
- On récupère cet objet qu'on peut mettre dans une **variable**

JS

HTML



<input>

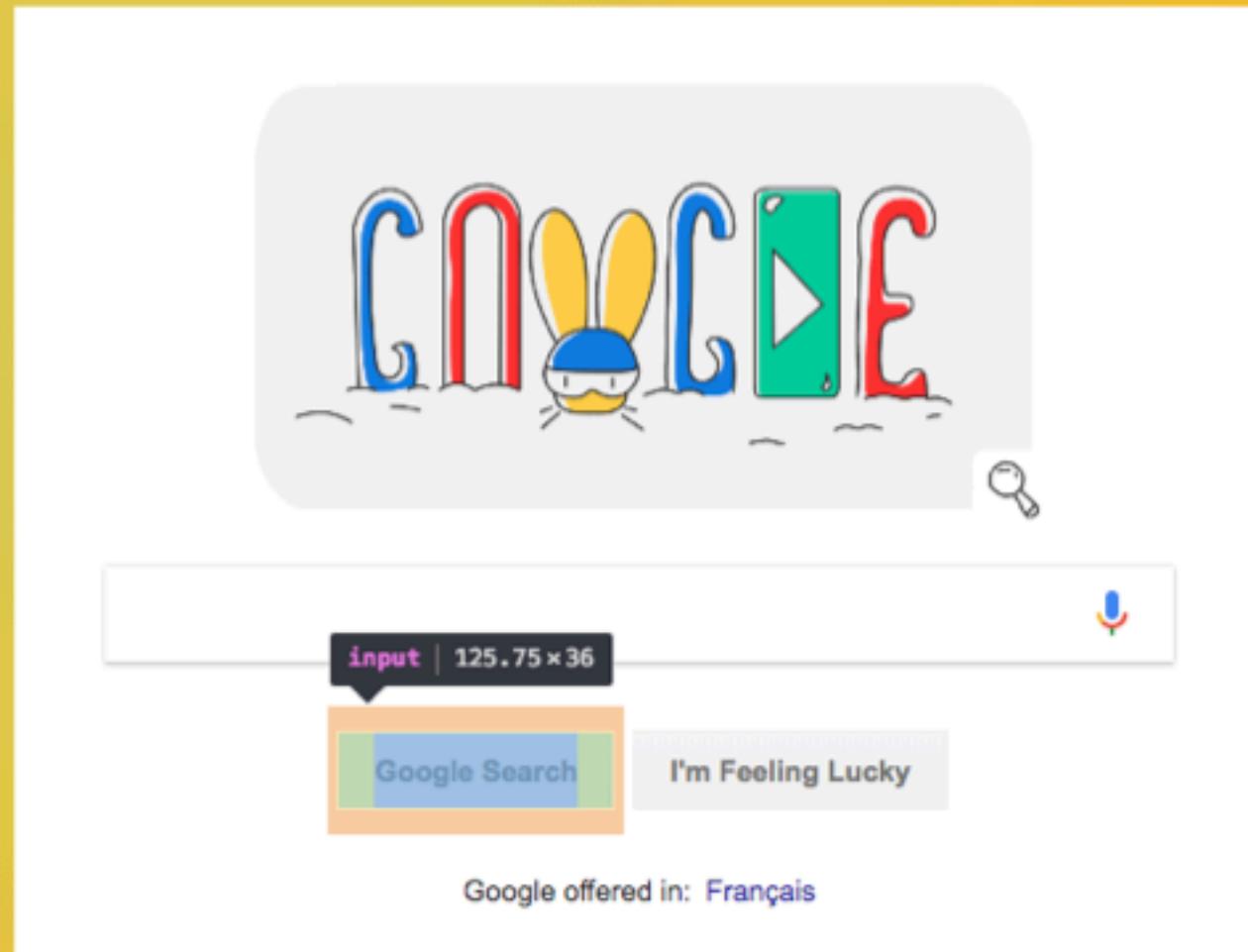
Sélectionner un élément

<button>

JS

Sélectionner un élément

- Je veux « attraper » un élément de ma page
- ici par exemple je veux récupérer ce bouton pour y accéder dans mon JavaScript



JavaScript dans le web

- Une **seule** méthode magique permet de sélectionner un élément du HTML pour l'utiliser en JavaScript :



```
document.querySelector("mon_element");
```

- On va la décortiquer
- ... mais ne l'apprenez pas encore par coeur ...

JS

JavaScript dans le web

```
document.querySelector("mon_element");
```

C'est **l'objet** qui représente la page Web



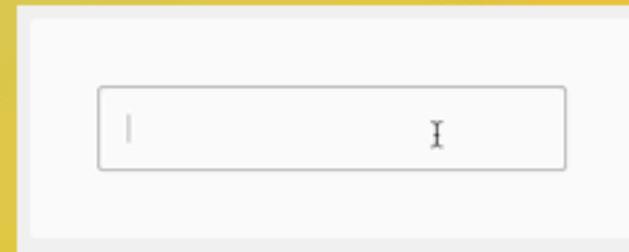
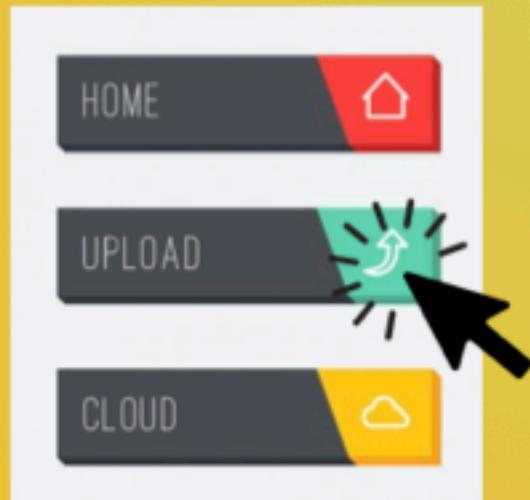
On invoque une fonction sur cet objet
(qu'on appelle une méthode)

On lui passe en argument l'élément que l'on recherche



JavaScript dans le web

- En fait, on aura pas de paramètres d'entrée sur le web, les paramètres d'entrée de votre code, ça sera les interactions de l'utilisateur avec la page web.
- Et c'est là que la magie opère



JS