Introduction à JavaScript

En savoir plus :

- http://emmanuel.coquery.pages.univ-lyon1.fr/enseignement/lifap5/
- https://grafikart.fr/formations/debuter-javascript

Objectifs du cours

- Comprendre les grands principes de la représentation de données et de documents numériques à l'aide d'un langage à balises
- Apprendre les bases d'HTML et CSS pour la génération de pages web
- Rendre des pages interactives avec JavaScript
- Mettre en ligne un site statique

Ce qu'on ne verra pas :

- La programmation côté serveur
- Les frameworks avancé côté client/navigateur

Plan global du cours

- Introduction aux langages à balises
  - Langage à balises
  - SGML
  - HTML
  - Aparté : le Web en 3 minutes
  - XML
- HTML
- CSS
- **JavaScript**

Aujourd'hui

- Rappels: HTML, CSS, JavaScript
- Bases: variables, conditions, boucles
- Fonctions, portée des variables et tapage
- Javascript dans un navigateur (Window, DOM, événements)
- Requêter un serveur (fetch)

Un exemple

```
<!DOCTYPE html>
<html>
    <head>
      <title>Une page HTML</title>
    </head>
    <body>
      <h1>Une page HTML</h1>
      <p>Ceci est une page HTML avec deux paragraphes.</p>
      <p>Dans ce paragraphe il y a un <a href=“https://
fr.wikipedia.org">lien vers wikipedia</a>.</p>
    </body>
</html>
```

Éléments d’une page web

Contenu - HTML

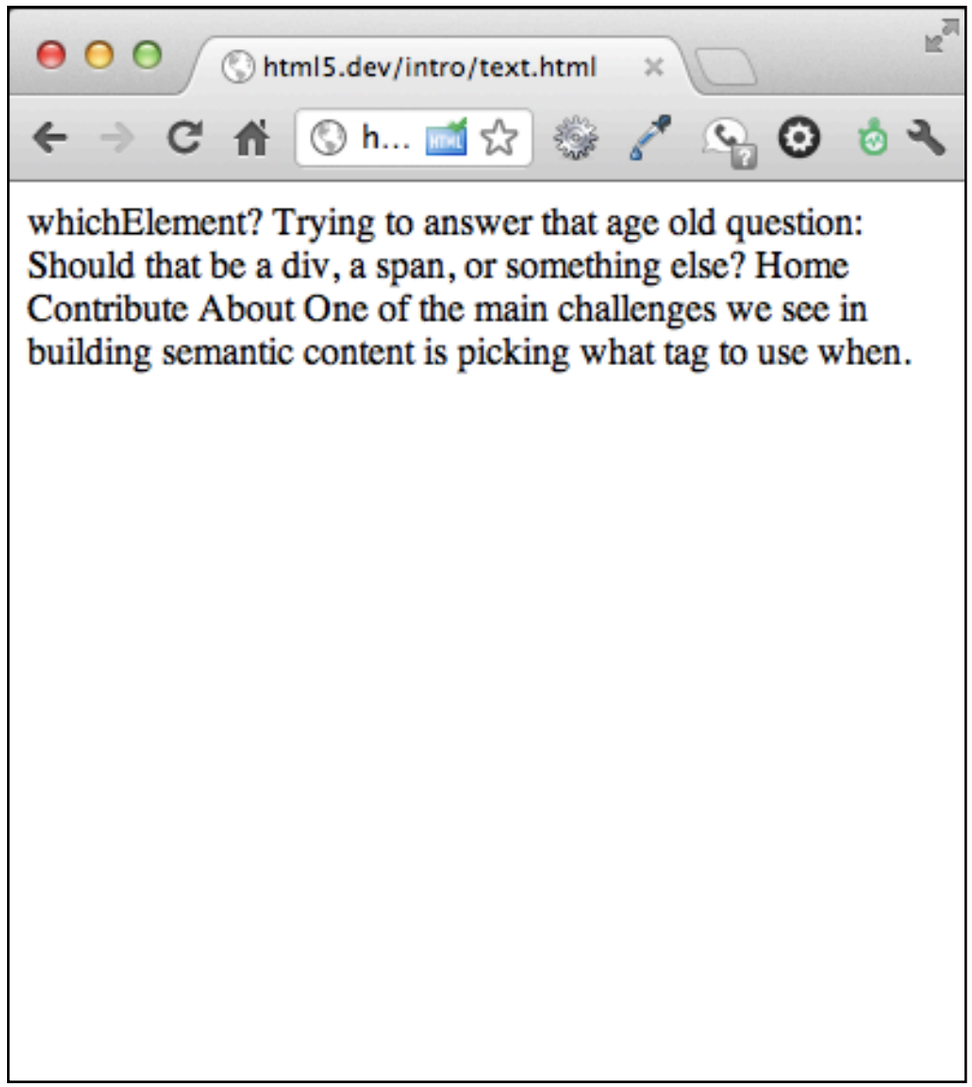
Structure - HTML

Style - CSS

Comportement - JavaScript

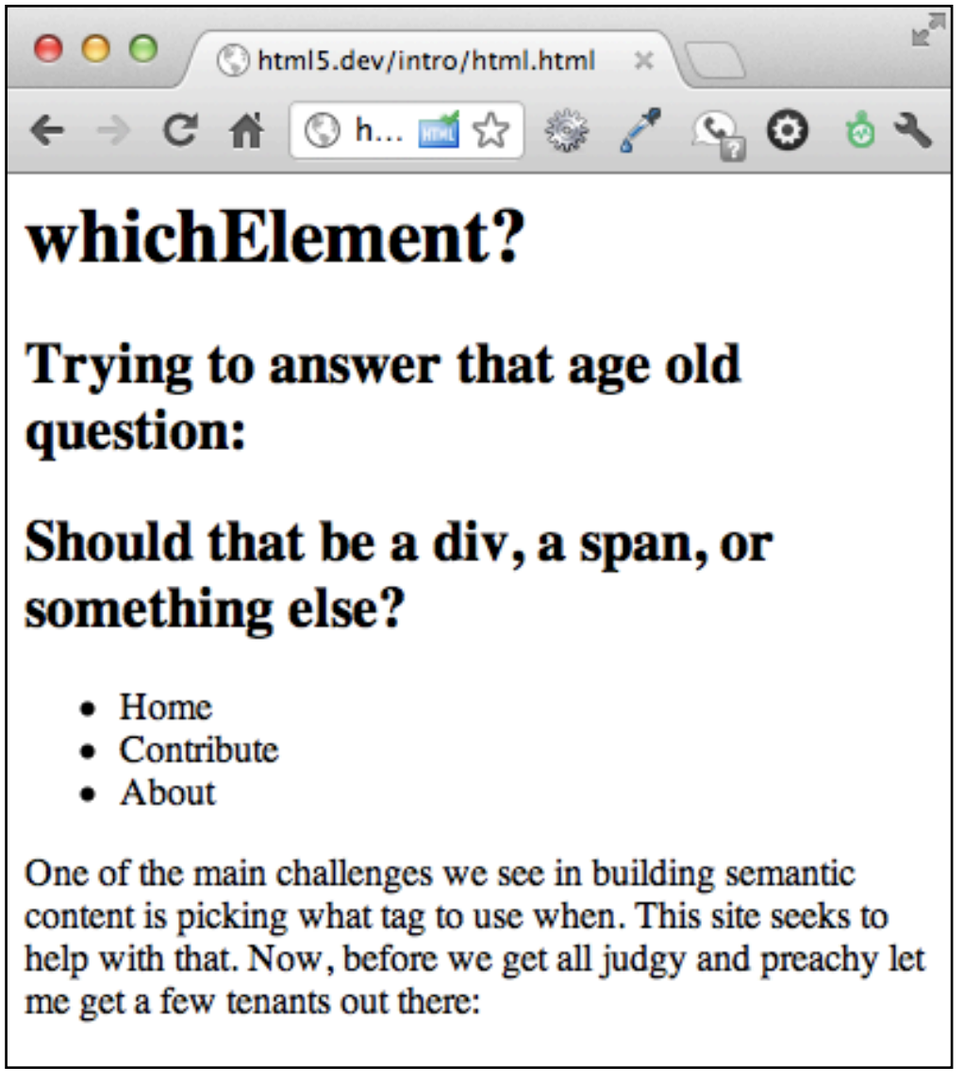
HTML : le contenu

```
whichElement?

Trying to answer that age old
question:

Should that be a div, a span,
or something else?



Home

Contribute

About


One of the main challenges we
see in building semantic
content is picking what tag to
use when.
```



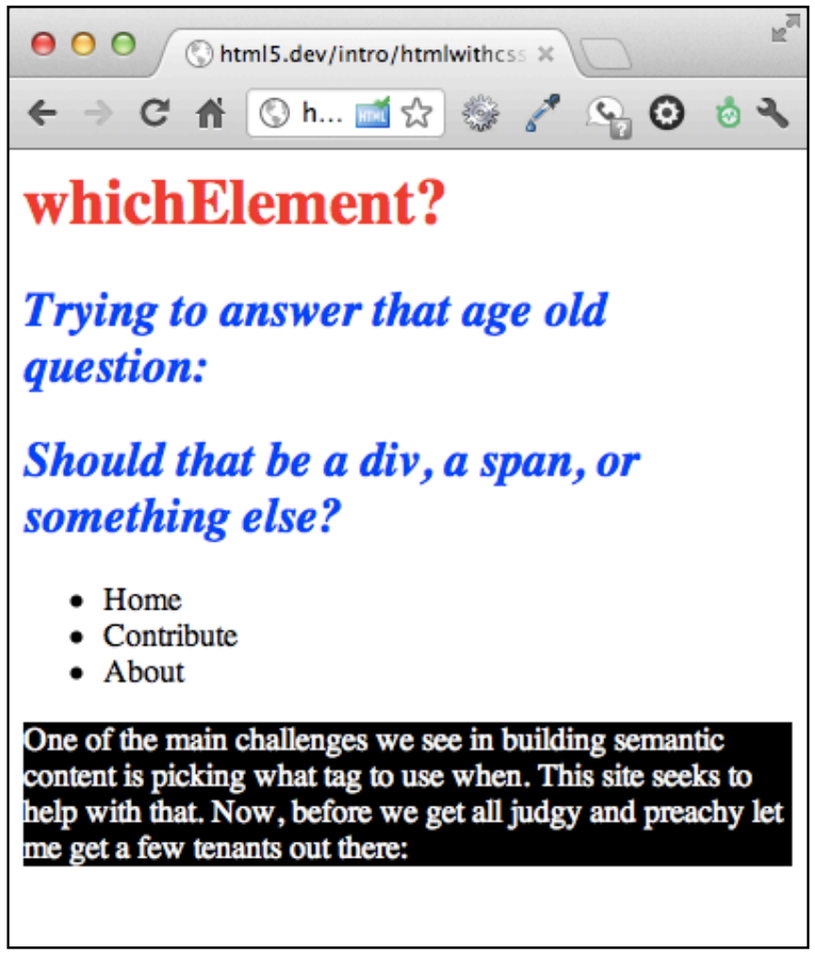
HTML : le contenu

```
<h1>whichElement?</h1>
<h2>Trying to answer that age
old question:</h2>
<h2>Should that be a div, a
span, or something else?</h2>

<ul>
<li>Home</li>
<li>Contribute</li>
<li>About</li>
</ul>

<p>One of the main challenges we
see in building semantic content
is picking what tag to use when.
This site seeks to help with
that. Now, before we get all
judgy and preachy let me get a
few tenants out there:</p>
```



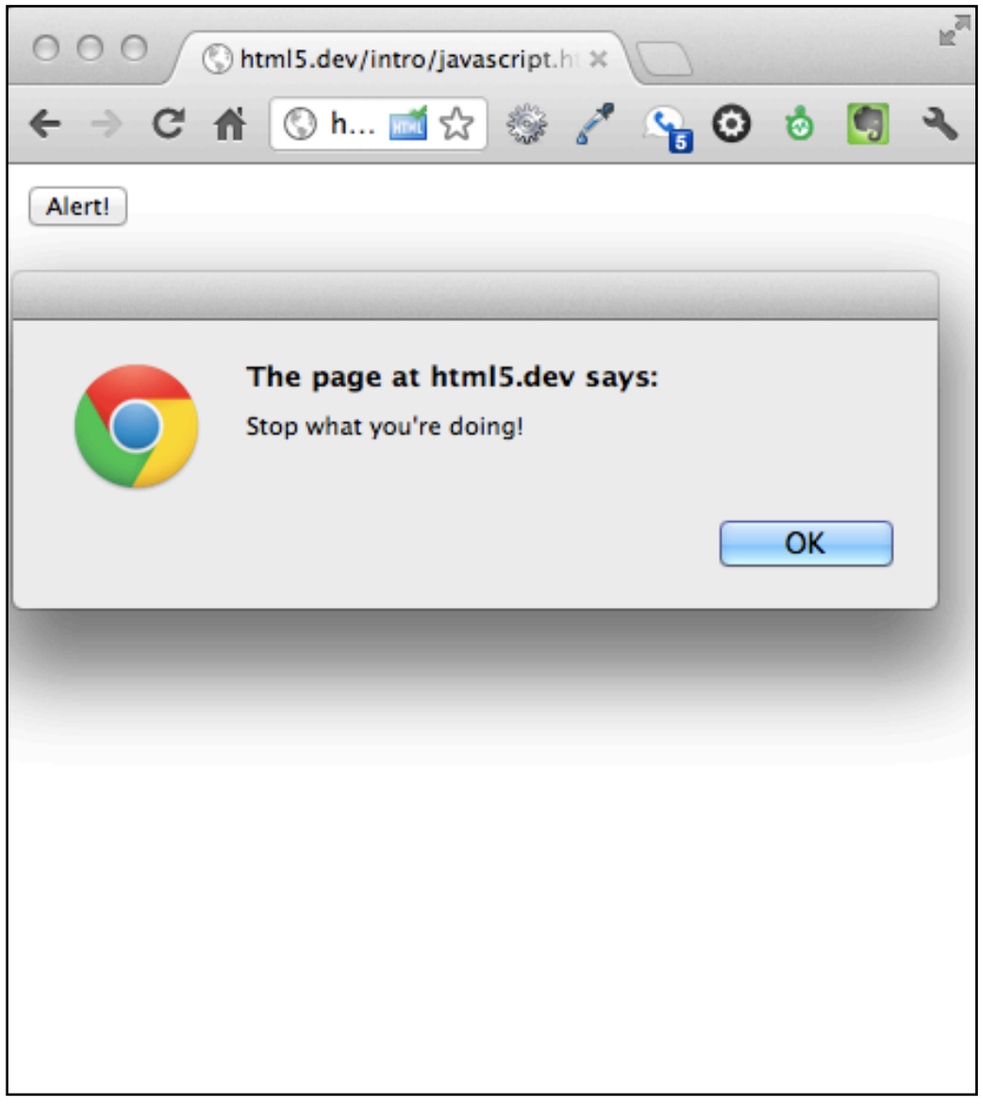
CSS : le style

```
h1{
   color: red;
}

h2{
   color: blue;
   font-style: italic;
}

p{
   color: white;
   background-color: black;
}
```



JavaScript

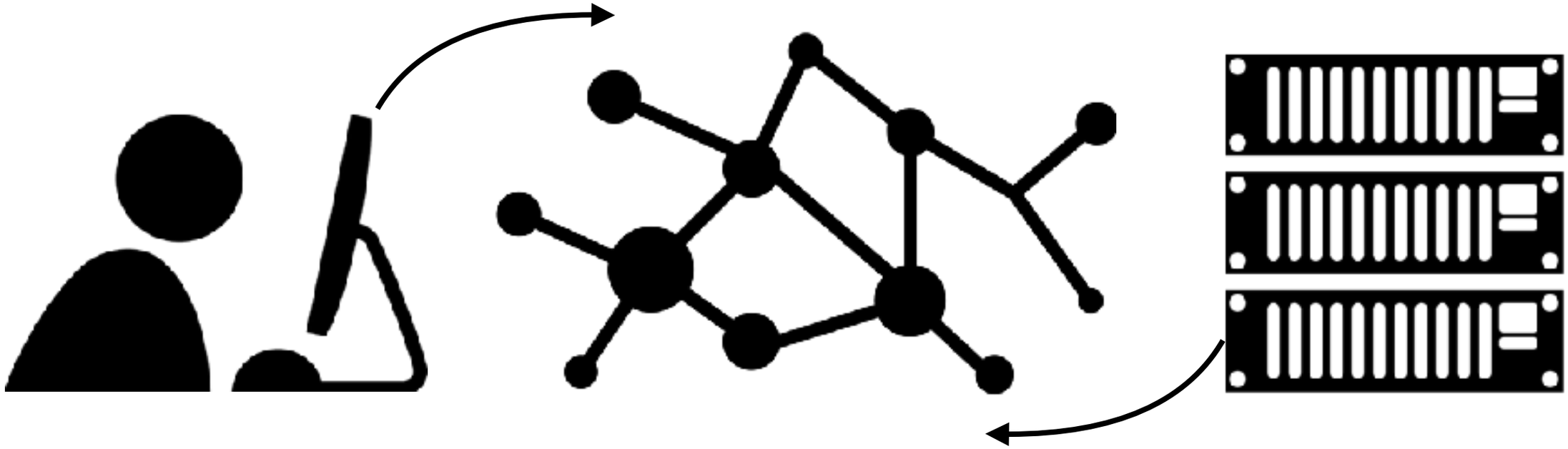
```
<html>

<head>
  <script type="text/javascript">
   function createAlertMessage(){
     alert("Stop what you're doing!");
   }
  </script>
</head>

<body>
  <button onclick="createAlertMessage()">
    Alert!
  </button>
</body>

</html>
```



Requête HTTP simple

1. Requête du navigateur web
GET /index.html HTTP/1.1

2. Le serveur web trouve le fichier
/var/www/…/index.html



3. Réponse du serveur
HTTP/1.x 200 OK
<contenu du fichier>

4. Le navigateur décompresse et affiche la page

Aujourd'hui

- Rappels: HTML, CSS, JavaScript
- **Bases: variables, conditions, boucles**
- Fonctions, portée des variables et tapage
- Javascript dans un navigateur (Window, DOM, événements)
- Requêter un serveur (fetch)

Les variables en JavaScript

Typage faible : le tapage est implicite et peut changer

- Nombres
- Chaines de caractères
- Booléens
- Tableaux
- Objets

https://codesandbox.io/live/45yxvoe

Conditions

```
if (booleen) {
    // code si vrai
} else {
    // code si faux
}
```

```
// condition ? <expression si vrai> : <expression si faux>
age = 19
"Je suis " + (age >= 18 ? "majeur" : "mineur")
```

```
switch (expression) {
  case valeur1:
    // Instructions à exécuter lorsque le résultat
    // de l'expression correspond à valeur1
    instructions1;
    break;
  case valeur2:
    // Instructions à exécuter lorsque le résultat
    // de l'expression correspond à valeur2
    break;
  default:
    // Instructions à exécuter lorsqu'aucune des valeurs
    // ne correspond
    break;
}
```

Boucles

Aller plus loin : https://hacks.mozilla.org/2015/04/es6-in-depth-iterators-and-the-for-of-loop/

```
var i = 0
while (i < 3) {
    "Je compte " + i
    if (i == 1) {
        break
    }
    i++
}
```

```
var i = 0
for (var i = 0; i < 3; i++) {
    "Je compte " + i
}
```

```
var eleves = ['Jean', 'Marc', 'Marie']
for (var i = 0; i < eleves.length; i++) {
    eleve[i] // vaudra alternativement : Jean, Marc, Marie
}
```

```
eleves.forEach(function (value) {
    console.log(value);
    // affichera alternativement : Jean, Marc, Marie
});
```

Aujourd'hui

- Rappels: HTML, CSS, JavaScript
- Bases: variables, conditions, boucles
- **Fonctions, portée des variables et tapage**
- Javascript dans un navigateur (Window, DOM, événements)
- Requêter un serveur (fetch)

Fonctions

```
function saluer (nom) {
    return "Salut " + nom
}
// On appel ensuite notre fonction avec
saluer('Marc') // Salut Marc
```

```
var convert_to_min = function (secondes)
  return secondes / 60
}

convert_to_min(3600) // 60
```

```
var eleve = {
    nom: 'Marc',
    note: 14,
    present: function () {
        return 'Je suis présent'
    }
}

eleve.present() // 'Je suis présent'
```

Portée des variables (var)

```
var a = function () {
    var b = 3
}
a()
b // ERREUR, b is not defined
```

```
var exterieur = "Salut";
var maFonction = function () {
  exterieur;
  var interieur = 3;
};
maFonction(); // Dans cet appel exterieur vaudra "Salut"
interieur; // ERREUR, interieur is not defined
```

```
var a = "Salut"; // portée globale
var maFonction = function () {
  var a = "demo"; // portée locale
  return a;
};
maFonction(); // demo
a; // Salut
// La variable a définie dans la fonction n'est pas la même
// que la variable définie dans la fonction
```

Portée des variables (let)

Introduit plus récemment pour limiter les bugs.

La portée de var n’est pas limitée :

```
for (var i = 0; i < 10; i++) {
    var maVar = true;
    console.log(maVar);
}
i; // 10
maVar; //true
```

Comme var, mais portée limitée au bloc courant.

```
function test(){
  if( 1 === 1 ){
    let maVar = true;
    console.log(maVar);
    // Retourne true
  }
  console.log(maVar);
  // Retourne une erreur, let n'existe pas hors du bloc "if"
}
```

Portée des constantes

Const : déclarer une constante disponible uniquement en lecture.

```
const CONFIG; // Retourne une erreur, il manque la valeur

const CONFIG = 'maConfig';
```

La portée de const est celle du bloc, comme la déclaration let. Pour la rendre globale, il faut simplement la définir hors de toute fonction.

```
function test(){
  if( 2 === 2 ){
    const CONFIG = 12;
    console.log(CONFIG);
    // Retourne 12
  }  console.log(CONFIG);
  // Retourne une erreur, config n'existe que dans le bloc courant
}
```

Console et débugging -> démo

ESLint

Principes du *linting* :

- Analyse statique du code (pas d'exécution)
- Erreurs probables, les mauvaise pratiques et éléments de style
- Configurables et intégrables aux IDE

Exemple de guide de style

- https://github.com/airbnb/javascript

Démo : https://codesandbox.io/s/demo-cci-4fisu?file=/src/variables.js

Aujourd'hui

- Rappels: HTML, CSS, JavaScript
- Bases: variables, conditions, boucles
- Fonctions, portée des variables et tapage
- **Javascript dans un navigateur**
- Requêter un serveur (fetch)

Window Object

https://developer.mozilla.org/en-US/docs/Web/API/Window

La fenêtre du navigateur

L'objet Window contient un ensemble de méthode et de propriétés

- largeur/hauteur de la fenêtre
- Location (url courant, historique)
- Alertes (pop-ups)
- Stockage de variables, de données
- Timers
- Accès aux capteurs

Document Object Model

Déjà vu au cours précédent (HTML/CSS)

Avec JavaScript on peut manipuler le DOM :

```
// Pour sélectionner un élément
document.body // Récupère l'élément body
document.getElementById('demo') // Sélectionne l'élément avec l'id demo
document.querySelector('.demo') // Sélectionne le premier élément correspondant au
sélecteur CSS

// Pour sélectionner plusieurs éléments
document.getElementsByClassName('demo') // Sélectionne les éléments avec la class demo
document.getElementsByTagName('p') // Sélectionne les éléments <p>
var elements = document.querySelectorAll('.demo') // Sélectionne les éléments
correspondant au sélecteur CSS
// Ces méthodes renvoient un objet NodeList enumerable
// On peut parcourir cette liste d'éléments comme un tableau
for (var i = 0; i < elements.length; ++i) {
    var element = elements[i] // objet de type Element
}
```

Programmation événementielle

1. Pour gérer l'interaction : clic, chargement de page, etc.
2. Pour gérer le parallélisme (un seul thread par page)

```
element.addEventListener("Type d'évènement", callback)
// Par exemple pour détecter un clic sur un lien

element.addEventListener('click', function () {
    window.alert('Vous avez cliqué sur le lien')
})
```

Aujourd'hui

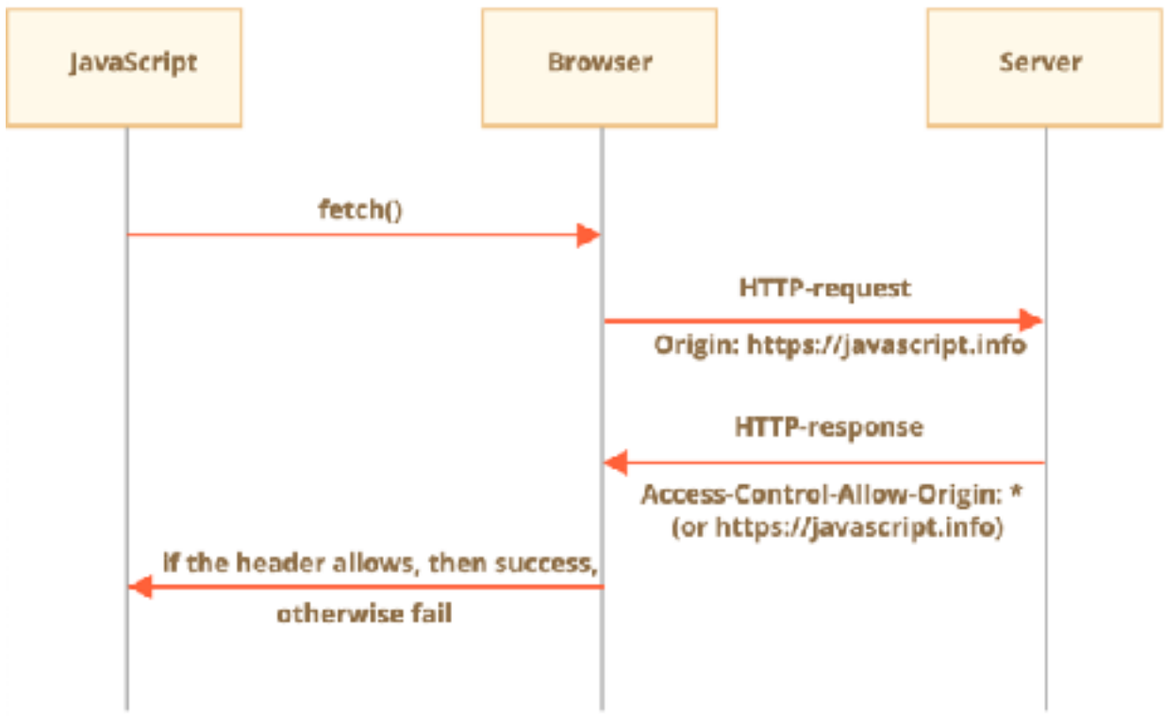
- Rappels: HTML, CSS, JavaScript
- Bases: variables, conditions, boucles
- Fonctions, portée des variables et tapage
- Javascript dans un navigateur (Window, DOM, événements)
- **Requêter un serveur (fetch)**

Fetch

En savoir plus :
- https://developer.mozilla.org/fr/docs/Web/API/Fetch_API/Using_Fetch
- https://grafikart.fr/tutoriels/fetch-1017#autoplay

Permet de faire une requête à un serveur, et de récupérer le contenu de manière asynchrone.

```
1 fetch('/api/v1/getsomedata')
2   .then(response => {
3     if (response.ok) {
4       return response.json();
5     } else {
6       console.log(response);
7     }
8 });
```



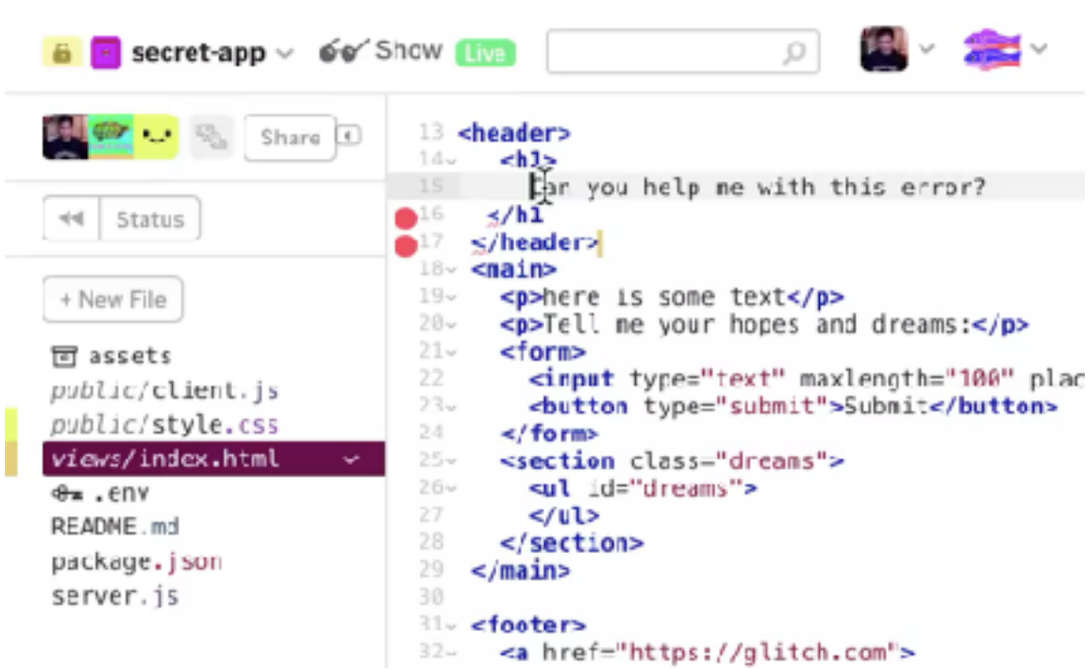
https://codesandbox.io/s/demo-cci-4fisu?file=/fetch1.html:0-723

Démo GitHub

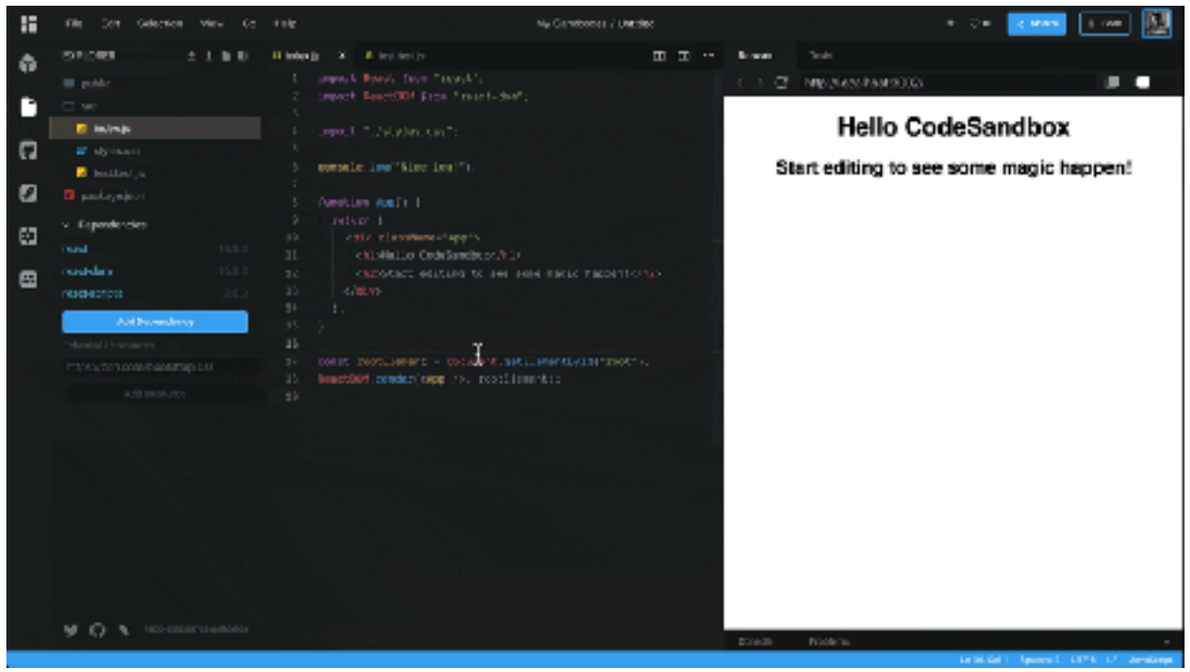
- Création d'un projet Github
- Clone du projet
- Commit, stage, push, pull, merge.
- Branches

- Issues
- Wiki
- Github pages

Alternatives au développement local



glitch.com



codesandbox.io

ES5 vs ES6

Les fonctions fléchées

```
// es5
var addition = function(x) {
    return x + 1;
};
// es6
let addition = (x) => {
    return x + 1;
}
// Avec un seul argument on peut se passer des parenthèses
let addition = x => {
    return x + 1;
};
// Comme il y a une seule instruction on peut aussi se passer des accolades
let addition = x => x + 1;
```

Les template strings

```
let personne = "Marie";
let message = `Bonjour les gens et bonjour ${personne}`;
console.log(message);

// Resultat : Bonjour les gens et bonjour Marie
```