



# Architecture(s) et application(s) Web

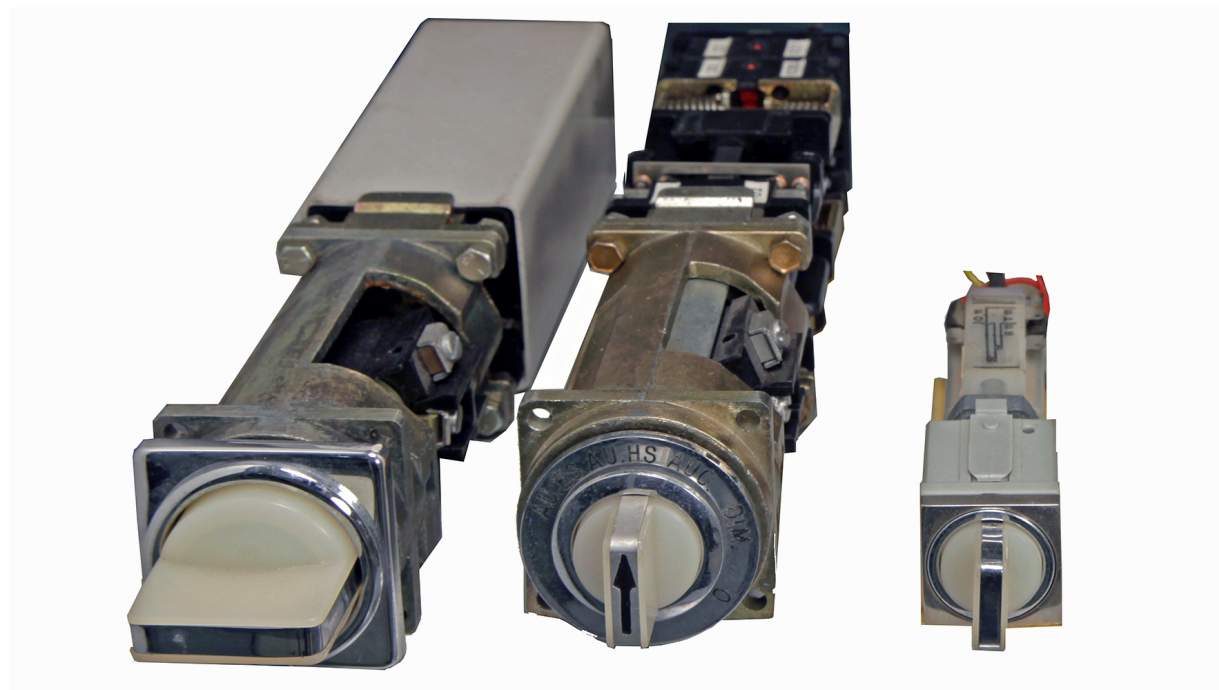
**CSC4101 - Expérience utilisateur Web**

04/09/2024

# Plan de la séquence

## 1. Interface utilisateur

# Interface homme-machine (IHM)



# Interface utilisateur

```
olivier@inf-11879: /home/olivier/git/fusionforge.int-evry.fr/appli-internet/appli-internet/CSC4101/symfony/agvoy
Fichier  Edition  Affichage  Rechercher  Terminal  Aide
olivier@inf-11879:~$ cd -
/home/olivier/git/fusionforge.int-evry.fr/appli-internet/appli-internet/CSC4101/symfony/agvoy
olivier@inf-11879:~/git/fusionforge.int-evry.fr/appli-internet/appli-internet/CSC4101/symfony/agvoy$ php bin/console server:run

[OK] Server running on http://127.0.0.1:8000

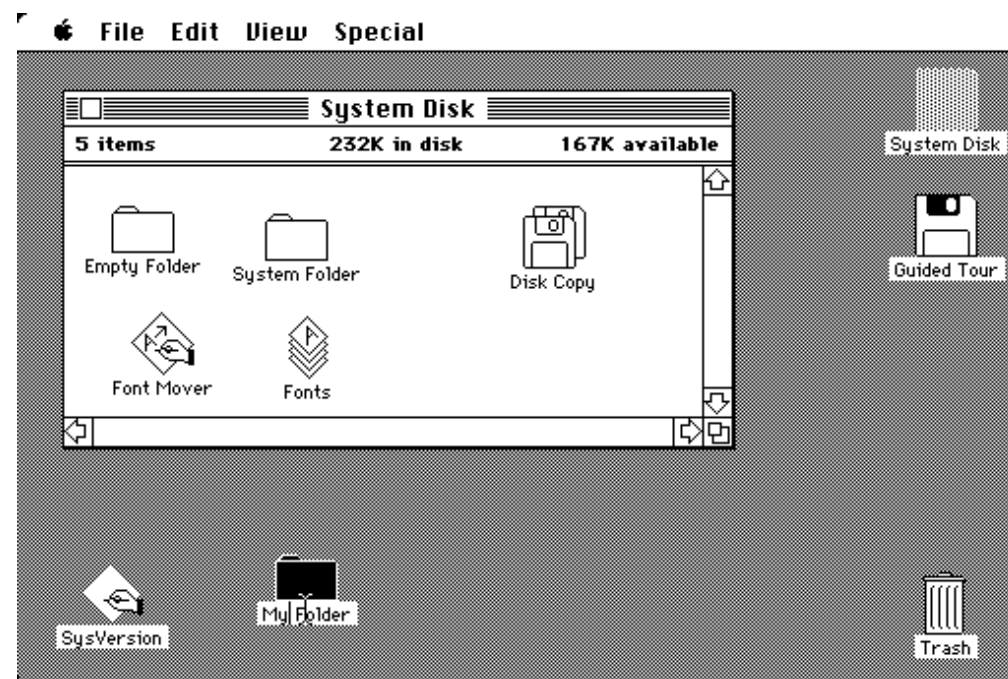
// Quit the server with CONTROL-C.
^C
olivier@inf-11879:~/git/fusionforge.int-evry.fr/appli-internet/appli-internet/CSC4101/symfony/agvoy$ ls
app  bin  composer.json  composer.lock  LICENSE  phpunit.xml.dist  README.md  src  tests  var  vendor  web
olivier@inf-11879:~/git/fusionforge.int-evry.fr/appli-internet/appli-internet/CSC4101/symfony/agvoy$ ls -l
total 136
drwxr-xr-x  4 olivier olivier 4096 mai  27 18:38 app
drwxr-xr-x  2 olivier olivier 4096 mai  24 17:19 bin
-rw-r--r--  1 olivier olivier 2868 mai  24 17:24 composer.json
-rw-r--r--  1 olivier olivier 91305 juil.  5 16:00 composer.lock
-rw-r--r--  1 olivier olivier 1065 mai  13 21:10 LICENSE
-rw-r--r--  1 olivier olivier  978 mai  13 21:10 phpunit.xml.dist
-rw-r--r--  1 olivier olivier 2285 mai  13 21:10 README.md
drwxr-xr-x  3 olivier olivier 4096 mai  13 21:10 src
drwxr-xr-x  3 olivier olivier 4096 mai  13 21:10 tests
drwxr-xr-x  5 olivier olivier 4096 mai  24 17:19 var
drwxr-xr-x 20 olivier olivier 4096 mai  24 17:19 vendor
drwxr-xr-x  3 olivier olivier 4096 mai  24 17:19 web
olivier@inf-11879:~/git/fusionforge.int-evry.fr/appli-internet/appli-internet/CSC4101/symfony/agvoy$
```

# Propriétés de la ligne de commande

*Command-Line Interface (CLI) :*

- Codifiée
- Stricte
- Statique

# Interface utilisateur



# Propriétés des interfaces graphiques

*Graphical User Interface* (GUI) :

- Métaphore (bureau)
- Exploratoire

# Interfaces de 0 à 77 ans





# Propriétés des interfaces « naturelles »

*Natural User Interface* (NUI) :

- Directe (passage de novice à expert facilement)
- Naturelle / Intuitive

# Importance des ordiphones (mobiles)

Plate-forme préférentielle (majorité des utilisateurs)

Utiliser technos Web pour applis mobiles :-)

`</interface_utilisateur>`

# Plan de la séquence

## 2. Interfaces Web

# Qualité des interfaces Web

# Ce qu'il ne faut pas faire

**CW NSFW** : <https://userinyerface.com/>

# Ergonomie

- **Expérience utilisateur** (*User eXperience : UX*)
- Utilisabilité :
  - apprenabilité (novices)
- « Fléau » de l'abandon de panier
- Utilisateurs « PIP » : *Pressés, Ignorants et Paresseux*

# Portabilité

Le Web est la plate-forme universelle.

- Standardisation = portabilité (merci HTML5)
- Applications mobiles :
  - Développé en HTML
  - « Compilé » en *toolkit* natif (ex. [Apache Cordova](#))
- **Attention** : prédominance de Chrome de Google... *Best viewed in Chrome*



# Accessibilité



Source : article [Comment les aveugles utilisent-ils internet ?](#) de *l'Obs*

# Obligations

- Règles pour l'accessibilité des contenus Web (WCAG) 2.1
  - Différents niveaux de priorité / niveaux de conformité à la norme
  - Principes : Perceptible, Utilisable, Compréhensible, Robuste
  - Règles, critères de succès
  - Techniques suffisantes et techniques recommandées
- Référentiel Général d'Amélioration de l'Accessibilité (RGAA)  
version 4.1.2 (avril 2023)

# Principes WCAG 2.0

# Principe 1 : perceptible

- 1.1 Proposer des **équivalents textuels** à tout contenu non textuel qui pourra alors être présenté sous d'autres formes selon les besoins de l'utilisateur : grands caractères, braille, synthèse vocale, symboles ou langage simplifié.
- 1.2 Proposer des versions de remplacement aux média temporels.
- 1.3 Créer un contenu qui puisse être présenté de différentes manières sans perte d'information ni de structure (par exemple avec une mise en page simplifiée)

# Principe 2 : utilisable

- 2.1 Rendre toutes les fonctionnalités **accessibles au clavier**.
- 2.2 Laisser à l'utilisateur suffisamment de temps pour lire et utiliser le contenu.
- 2.3 Ne pas concevoir de contenu susceptible de **provoquer des crises**.
- 2.4 Fournir à l'utilisateur des **éléments d'orientation** pour naviguer, trouver le contenu et se situer dans le site.

# Principe 3 : compréhensible

- 3.1 Rendre le contenu textuel lisible et compréhensible.
- 3.2 Faire en sorte que les pages apparaissent et fonctionnent **de manière prévisible**.
- 3.3 Aider l'utilisateur à éviter et à corriger les erreurs de saisie.

# Principe 4 : robuste

- 4.1 Optimiser la compatibilité avec les agents utilisateurs actuels et futurs, y compris avec les technologies d'assistance.

etc.

# La suite sur DesignGouv

*Concevons des services publics numériques accessibles, inclusifs et humains.*

<https://design.numerique.gouv.fr/>



# Évaluation qualité

- Critères Opquast (*Open Quality Standards*) :  
<https://checklists.opquast.com/fr/qualiteweb/>

`</qualite_interface>`

`</interfaces_web>`

# Plan de la séquence

## 3. Habillage des pages Web

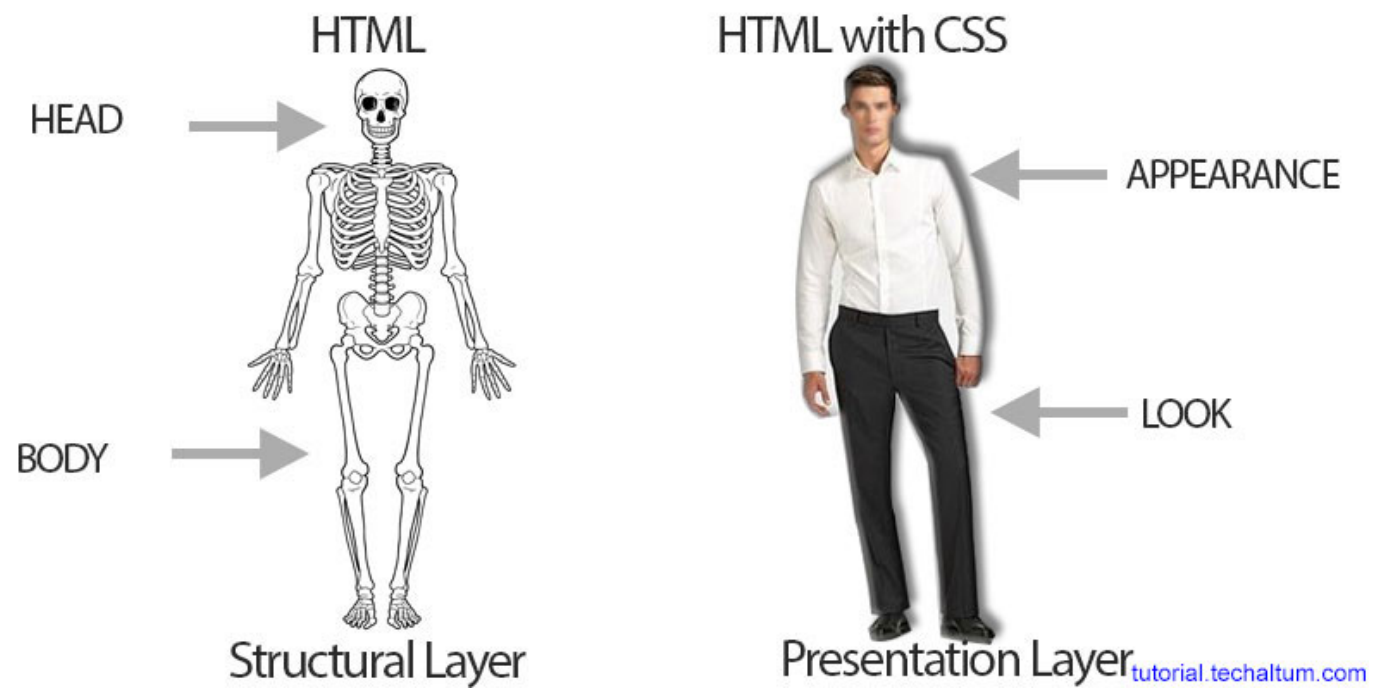
# Découplage structure / présentation

*(suite)*

# Page Web et document HTML

Conversion de **documents** en **une présentation** dans le navigateur.

# HTML + CSS



# Structure des sources des pages

Une « page Web » :

- un document HTML (maître)
- plus des images, boutons, menus, etc. (éventuellement externes)

Un document (HTML) :

- un arbre de rubriques,
- sous-rubriques,
- etc.

**Arbre de balises HTML (DOM : *Document Object Model*)**



# Structure d'une page affichée

Une page affichée en 2D (ou imprimée) :

- des boîtes qui contiennent d'autres boîtes
- boîtes qui contiennent texte ou images, etc.
- texte qui contient des caractères
- caractère qui contiennent des pixels
- ...

# Arbre DOM et rendu des pages

- Le navigateur (moteur de rendu) convertit :
  - Arbre DOM : *Document Object Model*en :
  - Arbre d'éléments à afficher
- **Règles de conversion**
  - Prédéfinies (navigateur)

# Exemple de page avec un tableau

Logo du site

Accueil

Lien

Lien

Lien

Lien

Lien

## Titre page

Une mise-en-page avec en-tête, barre de navigation et section de bas-de-page. Les lignes 1, 2 et 4 du tableau créent respectivement l'en-tête, la barre de navigation et le bas-de-page, et contiennent une seule cellule de tableau chacune.

La ligne 3 du tableau contient 3 cellules qui créent la colonne de menu (gauche), la colonne de contenu (milieu) et la colonne supplémentaire (droite).

Copyright ©

# Structure HTML correspondante ?

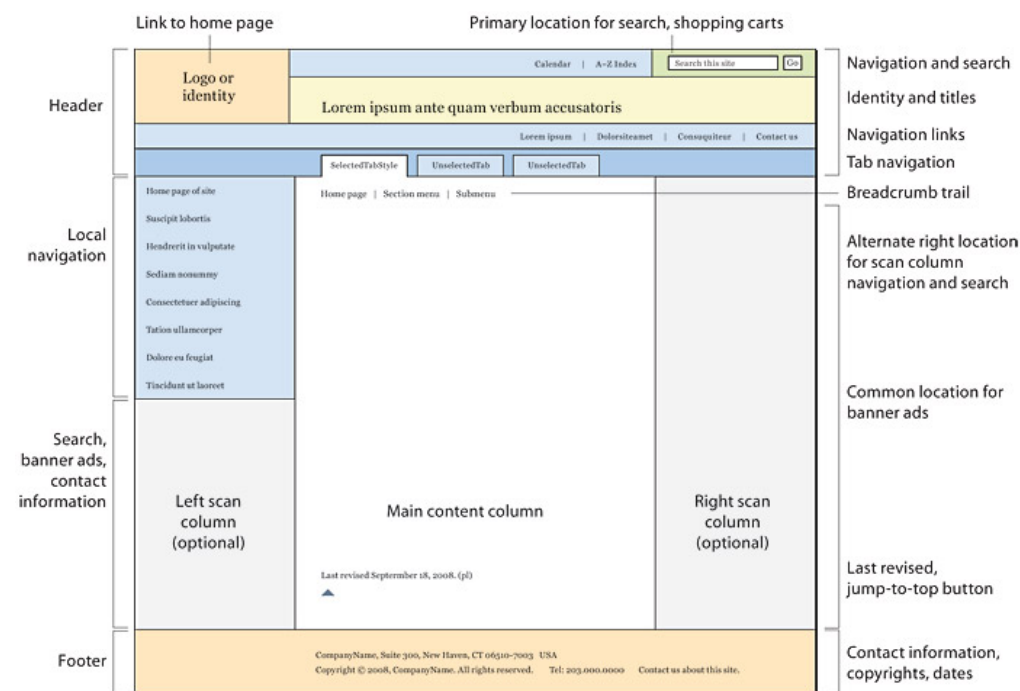
```
<table>
  <tr> <!-- SECTION EN-TETE  -->
    <td colspan="3">
      <h3>Logo du site</h3>
    </td>
  </tr>
  <tr> <!-- SECTION BARRE NAVIGATION == -->
    <td colspan="3">
      <a href="#">Accueil</a>
    </td>
  </tr>
  <tr>
    <td width="20%"> <!-- COLONNE GAUCHE (MENU) == -->
      <a href="#">Lien</a><br>
      <a href="#">Lien</a><br>
    </td>
    <td width="55%"> <!-- COLONNE MILIEU (CONTENU) == -->
      <h4>Titre page</h4>
      Une mise-en-page avec en-tête, barre de navigation et section de
      ...
    </td>
    <td width="25%">
```

# Faire mieux !

- Structurer la page en 6 **sections**, par exemple, avec des balises `<div>`
- Positionner le contenu de chaque section avec des règles CSS.
- Permet de faire évoluer la mise en page, par exemple sous forme linéaire et non plus tabulaire, etc.

# Programmer en HTML et CSS

# Structure archétypale d'une page



LOOK AT OUR COOL NEW THING

HEADE

SOME SUBSCRIPTION THING

THE STUFF YOU ACTUALLY CARE ABOUT

COOKIES

CHAT

JOIN SOME THING



# Exemples : *CSS Zen Garden*

<http://www.csszengarden.com/tr/francais/>

# RIP Molly Holzschlag



# Design

Outils de *Mockup* :

- Papier + crayon + gomme
- Outils ([penpot](#))
- HTML + CSS (+ *templates*)

# Guides de style

- Chercher « *web interface style guide* » sur votre moteur de recherche préféré
- Faire comme sur les applis mobiles :
  - Android : *Material design* :  
<https://developer.android.com/design/index.html>
  - Apple : *Designing for iOS* :  
<https://developer.apple.com/design/human-interface-guidelines/designing-for-ios>

# Inspiration / prêt à l'emploi

- <https://www.awwwards.com/> : récompenses pour le *Web design*
- <http://www.webdesign-inspiration.com/> : sélection de *designs*

Marché :

- par ex.: <https://themeforest.net/> : thèmes prêts à l'emploi
- *plein d'autres*

# Caractéristiques de HTML

# HTML 5

- Tout (ou presque) est possible en terme d'interface d'applications.
- Y compris l'accès à des zones de graphisme libre et l'intégration des périphériques du téléphone/ordinateur

# Les bons artistes copient, les grands artistes volent

Avantage d'HTML : le code source HTML dispo.

Vive le copier-coller :-)



# Contenu adaptatif

- *Responsive design* : prend en compte la taille de l'écran automatiquement
- p. ex.: avec [Bootstrap](#) (voir plus loin)

# Interactions dynamiques

- Javascript
- Interactions asynchrones

# Interactions avec ordiphone mobile

- Application allant plus loin que l'affichage et la saisie de texte : accès à toutes les fonctions du terminal mobile depuis le navigateur Web

[Roadmap of Web Applications on Mobile](#) du W3C (September 2020)

# Principes de CSS

*Cascading Style Sheets* (Feuilles de style de Cascade)

Concevoir l'habillage des pages du site / application

# Cascade

Modèle « objet » particulier

- Langage déclaratif
- Combinaison de plusieurs feuilles avec priorités
- « Héritage » de propriétés des parents
- Factorisation de motifs :

**DRY** – *don't repeat yourself*

# Langage à base de règles

- **Si** motif trouvé (sélecteur)
- **alors** valeur donnée à attribut/propriété de mise en forme

h1 {color: green}

sélecteur    propriété    valeur

# Exemple

```
h1 {  
  font-size: 60px;  
  text-align: center;  
}  
  
p,  
li {  
  font-size: 16px;  
  line-height: 2;  
  letter-spacing: 1px;  
}
```

# Niveaux croissants de proximité

Emplacement du code CSS :

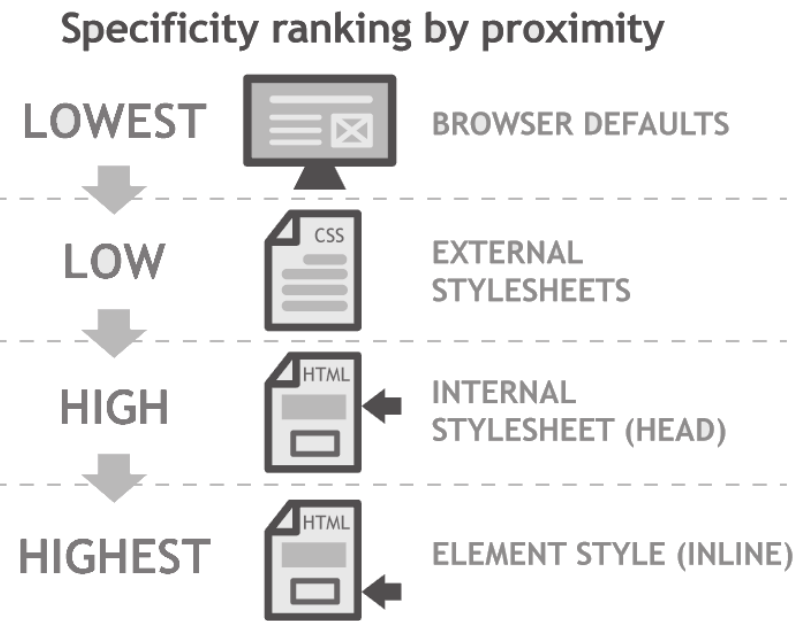
1. pas de style : affichage navigateur par défaut
2. fichier CSS externe :

```
<link rel="stylesheet" href="external.css"/>
```

3. bloc style interne :

```
<style type="text/css">  
.underline { text-decoration: underline; }  
</style>
```





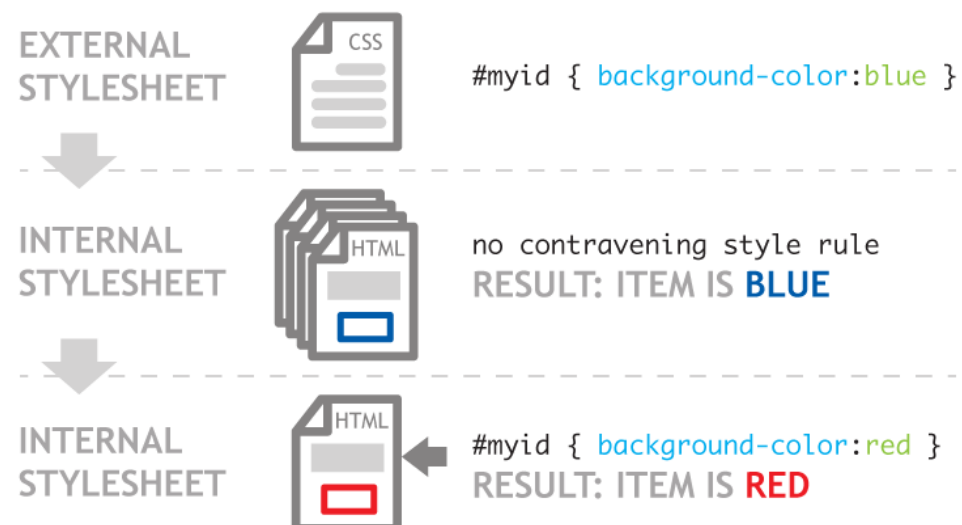
Ernie Simpson, aka The Big Erns

Figure 5 : Specificity ranking by proximity

# Spécificité selon la proximité

Quelles règles s'appliquent, si plusieurs définies ?

## Proximal Specificity in action...



# Spécificité du sélecteur

Ordre croissant de spécificité du sélecteur :

1. par balise : `div...{style:valeur}`.
2. par classe : `..myclass...{style:valeur}`.
3. par identifiant : `#myid...{style:valeur}`.

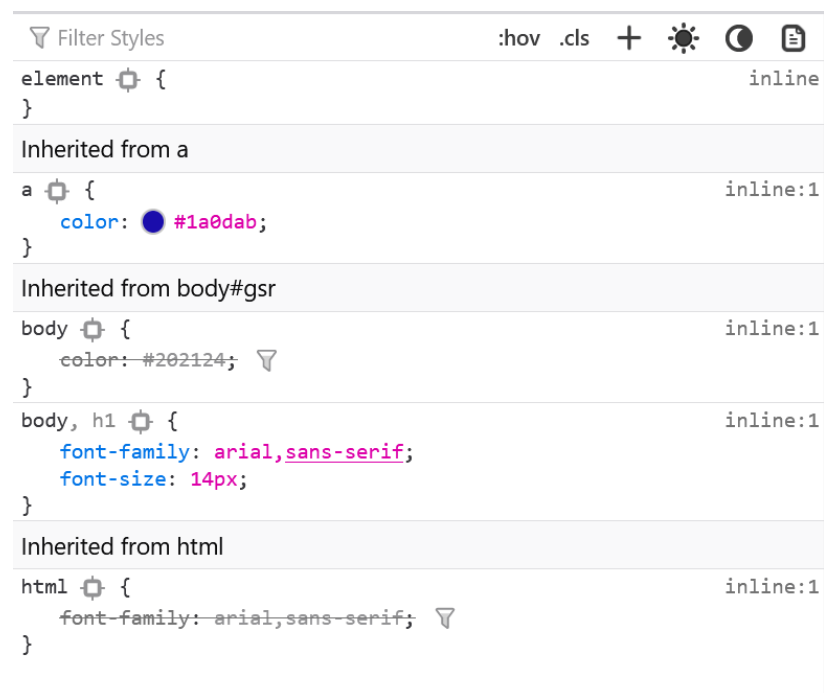
## Specificity ranking by Selector type



Ernie Simpson, aka The Big Erns

Figure 7 : Specificity ranking by selector type

# Outils du développeur



The image shows a browser's developer tools style inspector. At the top, there is a search bar with the text "Filter Styles" and a dropdown menu showing ":hov .cls". To the right of the search bar are icons for a plus sign, a sun (representing light mode), a moon (representing dark mode), and a document icon. Below the search bar, the style inspector displays a list of CSS rules. The first rule is for the "element" selector, which is currently selected and shows "color: #1a0dab;". Below this, there are sections for "Inherited from a", "Inherited from body#gsr", "Inherited from body, h1", and "Inherited from html". Each section shows the CSS rule for that selector, including the "color" property and its value. The "color" property is highlighted in blue in the original image.

```
Filter Styles :hov .cls + ☀ 🌙 📄  
element { inline  
}  
Inherited from a  
a { inline:1  
  color: #1a0dab;  
}  
Inherited from body#gsr  
body { inline:1  
  color: #202124;  
}  
body, h1 { inline:1  
  font-family: arial,sans-serif;  
  font-size: 14px;  
}  
Inherited from html  
html { inline:1  
  font-family: arial,sans-serif;  
}
```

# Conception de CSS

- Pas toujours simple
- Abstraction
- *Frameworks* CSS (et Javascript)
  - ex. : [Bootstrap](#)

Couplage blocs *templates* Twig et CSS

</habillage\_css>

# Plan de la séquence

## 5. Conception blocs Twig et CSS - Bootstrap



# Conception des gabarits et CSS

# Concevoir les gabarits des pages

Conception parallèle :

- des blocs des gabarits HTML
- des sections `<div>` qu'ils contiennent (identifiants ou classes)
- des mises en forme

Même convention de nommage

# Modèle de page souhaité

```
<html>  
  <head>  
  </head>  
  <body>  
    NAVIGATION  
  
    MENU (COLONNE DE GAUCHE)  
  
    CONTENU PRINCIPAL  
  
    BAS DE PAGE  
  </body>  
</html>
```

# Structure sémantique

```
<body>
  <div id="navigation">
    (NAVIGATION)
  </div>
  <div id="menu">
    (MENU RUBRIQUES)
  </div>
  <div id="main">
    (CONTENU PRINCIPAL)
  </div>
  <div id="footer">
    (BAS DE PAGE)
  </div>
</body>
```

# Gabarit Twig

```
{% block body %}
<body>
  <div id="navigation">
    {% block navigation %}
    {% endblock %} {# navigation #}
  </div>
  <div id="menu">
    {% block menu %}
    {% endblock %} {# menu #}
  </div>
  <div id="main">
    {% block main %}
      <h1>{{ Message }}</h1>
    {% endblock %} {# main #}
  </div>
  <div id="footer">
    {% block footer %}
    {% endblock %} {# footer #}
  </div>
</body>
{% endblock %} {# body #}
```

# Standardisation / Spécialisation

Spécialisation dans une sous-section ou une page de certains éléments de structure ou de présentation

- héritage des gabarits : redéfinition du contenu de blocs

```
{% block main %}  
...  
{% endblock %}
```

- cascade des feuilles de style CSS : redéfinition de la mise en forme

```
#main { background-color: lightblue; }
```

</conception>

# CSS avec Bootstrap



# Utilisation d'un *framework* de présentation CSS

- Standardisation du look
- Adaptatif
- Grille pour le positionnement graphique
- Intégration avec Twig / Symfony

# Bootstrap

*Framework CSS (+ JS)*

<http://getbootstrap.com/>



# Systeme de grille

Mise en page basée sur une grille de 12 colonnes

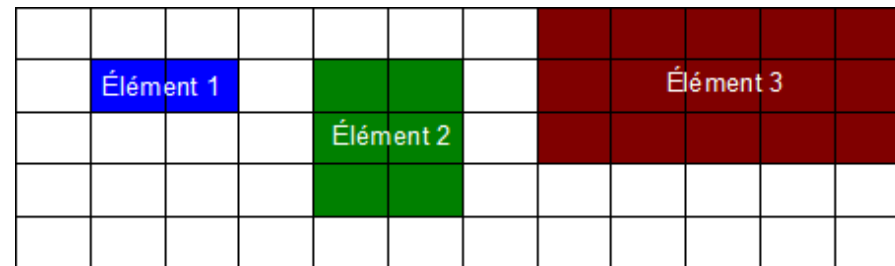


Figure 9 : Grille de mise en page dans Bootstrap

# Exemples

- exemples, dans la documentation Bootstrap
- exemples sur *Start Bootstrap*

# Thèmes

- Thèmes sur *Start Bootstrap*
- Feuille de style additionnelle

`</bootstrap>`

</technos\_projet>

# *Take away*

- Structure != présentation
- Rôle de CSS
- Principe de surcharge des feuilles (cascade)
- Accessibilité
- Bootstrap



# Récap

- HTTP (GET)
- PHP
- Doctrine
- Routeur Symfony
- HTML
- Twig
- **CSS**

**Ensuite...**

# Générateur Contrôleur CRUD Symfony

Pour vos projets, mieux que `make...controller` :

```
symfony console make:crud Todo
```

Génère code + gabarits :

```
created: src/Controller/ToDoController.php
created: src/Form/ToDoType.php
created: templates/todo/_delete_form.html.twig
created: templates/todo/_form.html.twig
created: templates/todo/index.html.twig
created: templates/todo/show.html.twig

created: templates/todo/new.html.twig
created: templates/todo/edit.html.twig
```

*spoiler* : formulaires, mais du temps gagné

# Postface

# Crédits illustrations et vidéos

- Illustration Desktop Macintosh source [Wikipedia](#)
- Vidéo <https://www.youtube.com/watch?v=gc9NpYrbZgQ> par [UserExperiencesWorks](#)
- vidéo : [Comment les aveugles utilisent-ils internet ?](#) (Le nouvel obs)
- Image « HTML + CSS » empruntée au « CSS Tutorial » de Avinash Malhotra : <http://tutorial.techaltum.com/css.html>
- Diagrammes « *Specificity ranking by proximity* », « *Proximal specificity in action...* » et « *Specificity ranking by selector type* » empruntés à par Ernie Simpson, *aka The Big Erns* : <http://cssway.thebigerns.com/special/master-item-styles/> (site disparu)
- Diagramme de grille de mise en page de Bootstrap empruntée à [Bootstrap de Twitter : un kit CSS et plus !](#) par Maurice Chavelli
- Illustration « [Enjoy Websurfing](#) », supposément de Bruce Sterling

# Copyright

- Document propriété de ses auteurs et de Télécom SudParis (sauf exceptions explicitement mentionnées).
- Réservé à l'utilisation pour la formation initiale à Télécom SudParis.