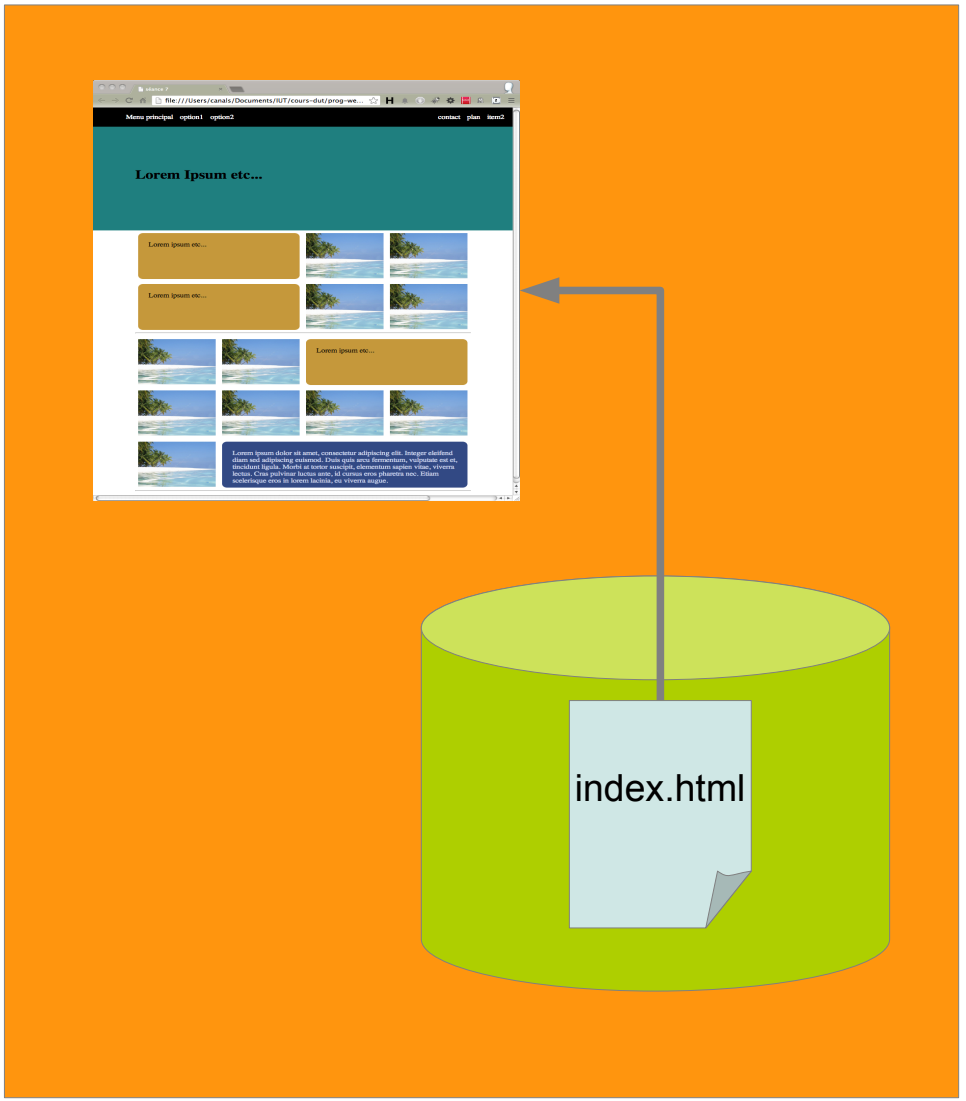
Développement d'interfaces Web
Module R1-02

BUT S1
Cours 7

Fonctionnement client-serveur

- Ce que vous avez fait jusqu'à présent
  - navigateur et document sur la même machine
  - chargement du fichier directement dans le navigateur
- Problème : le document n'est pas accessible sur d'autres machines



- **Fonctionnement en client/serveur : mettre les documents à disposition de machines distantes**
  - le navigateur utilisé pour la visualisation et les documents HTML/CSS sont sur des machines différentes
  - nécessite une connexion réseau entre les 2 machines
- **Nécessite un *protocole* de dialogue entre les 2 machines**
  - elles doivent échanger de façon compréhensible et donc en utilisant le même *langage* : HTTP
  - les navigateurs « parlent » HTTP : ce sont des clients HTTP
  - la machine distante détenant les documents HTML doit disposer d'un logiciel spécialisé : un serveur HTTP (Web)

machine cliente

machine serveur

Requête HTTP :
demande d'accès
à un document

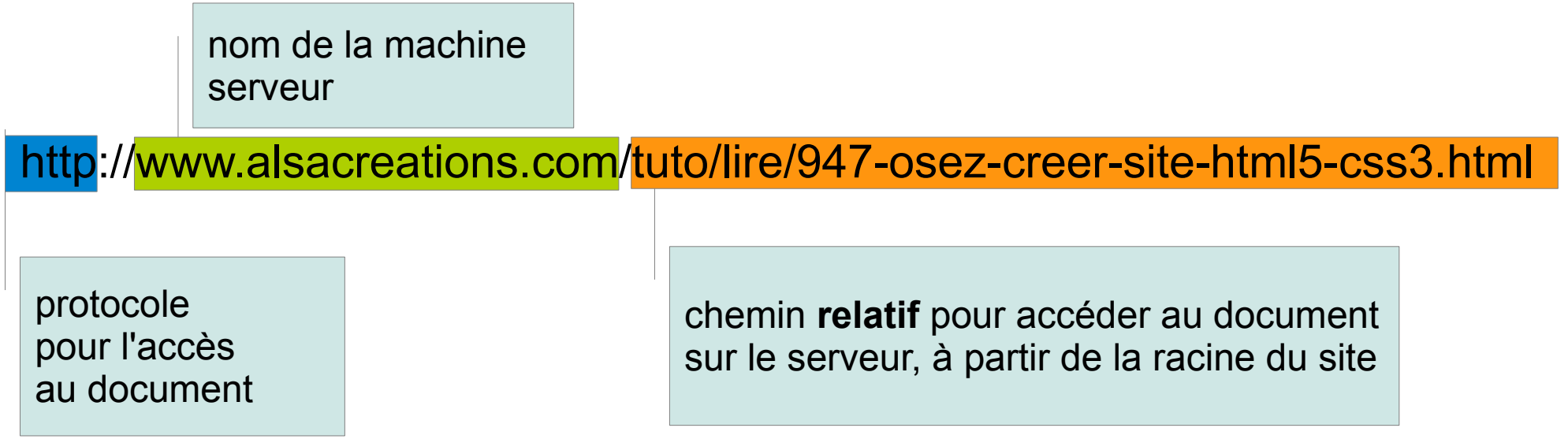
serveur HTTP

Réponse HTTP :
contenu du
document (HTML)

index.html

Les URL

- *Uniform Ressource Locator*
- Mécanisme de désignation et de localisation des documents sur internet
- Permet de désigner et d'accéder à un document à partir de n'importe qu'elle machine
- Anatomie d'une URL :



- **Rôle du client**
  - identifier la machine serveur
  - lui transmettre une requête HTTP en fournissant le chemin relatif
  - recevoir la réponse HTTP et le texte HTML
  - afficher le document HTML mis en forme

- **Rôle du serveur**
  - recevoir la requête HTTP
  - accéder au document à l'aide du chemin relatif, à partir de la racine du site
  - transmettre au client une réponse HTTP contenant le texte HTML du document

Les URL

- Quelques règles pour constituer les URL, et donc pour **nommer les répertoires et documents** sur le serveur
  - éviter les espaces et caractères accentués
  - ne pas utiliser les caractères suivants :

    : / ? # [ ] @ ! $ & ' ( ) * + , ; =

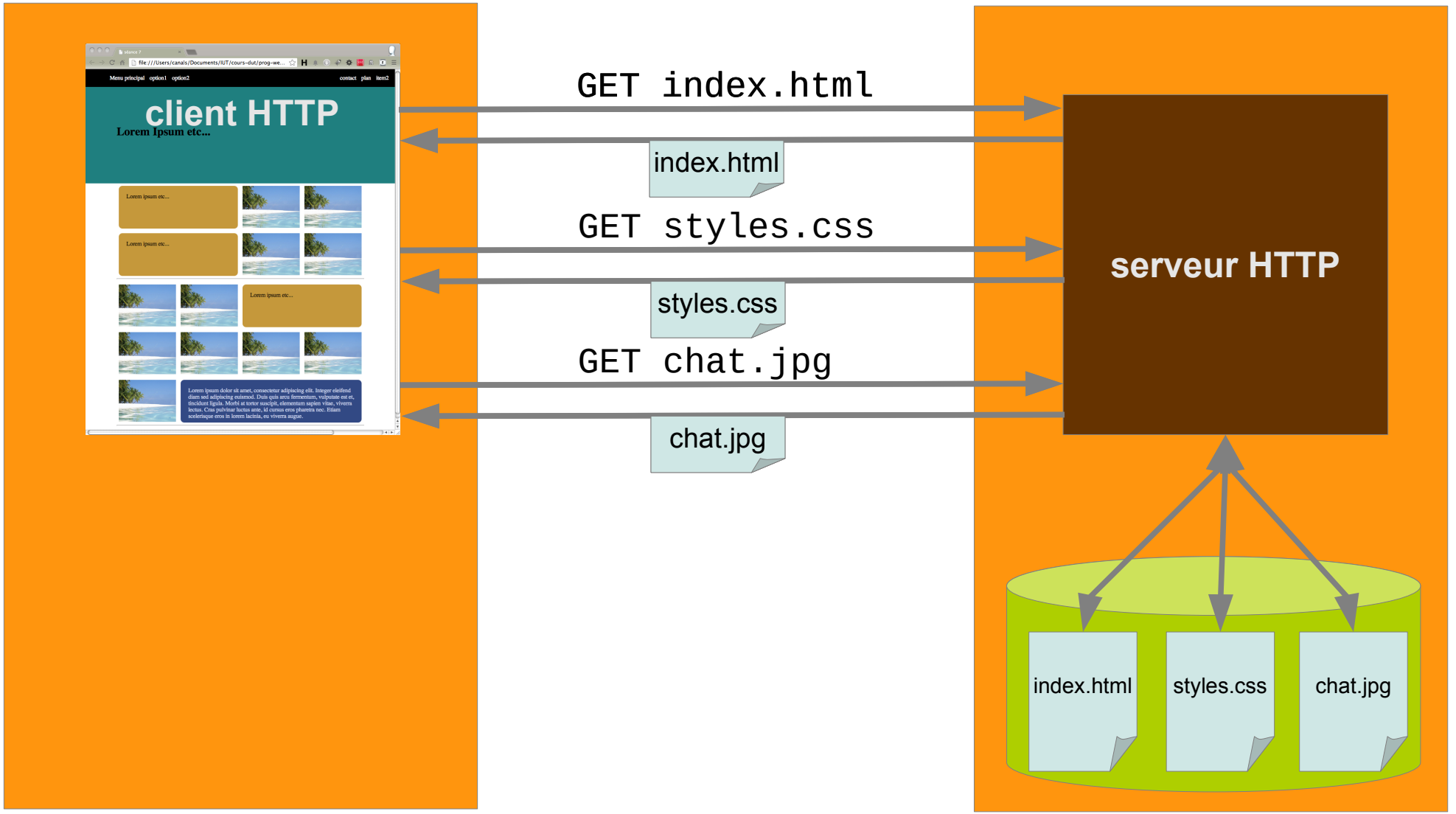
Zoom sur le chargement complet d'une page

- Le client (navigateur) envoie la requête de l'URL de départ et récupère le document HTML
- Le document est analysé pour extraire tous les fichiers externes nécessaires (images, CSS…)
- Chaque fichier externe fait à son tour l'objet d'une requête vers le serveur
- Le chargement complet d'une page implique ainsi *n* requêtes vers le serveur...

Zoom sur le chargement complet d'une page

machine cliente

machine serveur



Exercice

- Chaque étudiant dispose d'un espace situé sous la racine des documents d'un serveur web
- URL: `https://webetu.iutnc.univ-lorraine.fr/www/<login>`
- Racine des documents : (volume) `W:\`
- => le document `W:\doc.html` est accessible avec l'URL :

`https://webetu.iutnc.univ-lorraine.fr/www/<login>/doc.html`

- Exemple

`https://webetu.iutnc.univ-lorraine.fr/www/dosch5/doc.html`

- Exercice : rendre tous vos exercices accessibles via le serveur HTTP installé sur la machine webetu

Autres techniques de positionnement

- Les boîtes flexibles : `flex`
- **Le positionnement basé grille : `CSS grid`**

CSS grid

- Modèle de disposition en grille (espace 2D, là où `flex` est plutôt orienté 1D)
- Basé sur un découpage en *régions*, organisées entre elles et pouvant *si nécessaire* se chevaucher
- Environnement de gestion spatiale plus riche encore que `flex`

Principes d'utilisation de CSS grid

1. Utilisation de `display: grid` dans un élément *container*
2. Définition de la structure
   - les colonnes : `grid-templates-columns`
   - les lignes : `grid-templates-rows`
3. Définition et positionnement des éléments enfants **directs** (les *items*)

Définition des dimensions de grilles

- Plusieurs syntaxes (exemples similaires pour les colonnes)
  - `grid-template-rows: 50px 100px 50px;`
  - `grid-template-rows: 50px 30% 20% auto;`
  - `grid-template-rows: [r1] 25% [r2] 300px;`
  - `grid-template-rows: 200px 1fr 2fr;`
  - `grid-template-rows: repeat(3, 30px 10%);`

3 lignes, tailles absolues

4 lignes, dernière en taille auto

2 lignes nommées (r1, r2)

2ième ligne occupant 1 fr(action) de l'espace restant, 3ième ligne occupant 2 fr(actions)

6 lignes : 3 répétitions de 2 lignes (une abs, une rel)

Disposition des items

- Plusieurs propriétés d'items pour l'emplacement
  - `grid-column-start` : colonne de départ
  - `grid-column-end` : colonne de fin
  - `grid-row-start` : ligne de départ
  - `grid-row-end` : ligne de fin
- Valeurs possibles
  - n° de ligne (resp. colonne) ou nom (si nommé) : définit une borne (**exclue** si de « fin »)
  - `span` *num* : nombre d'emplacements occupés
  - `span` *nom* : prolongement jusqu'au nom

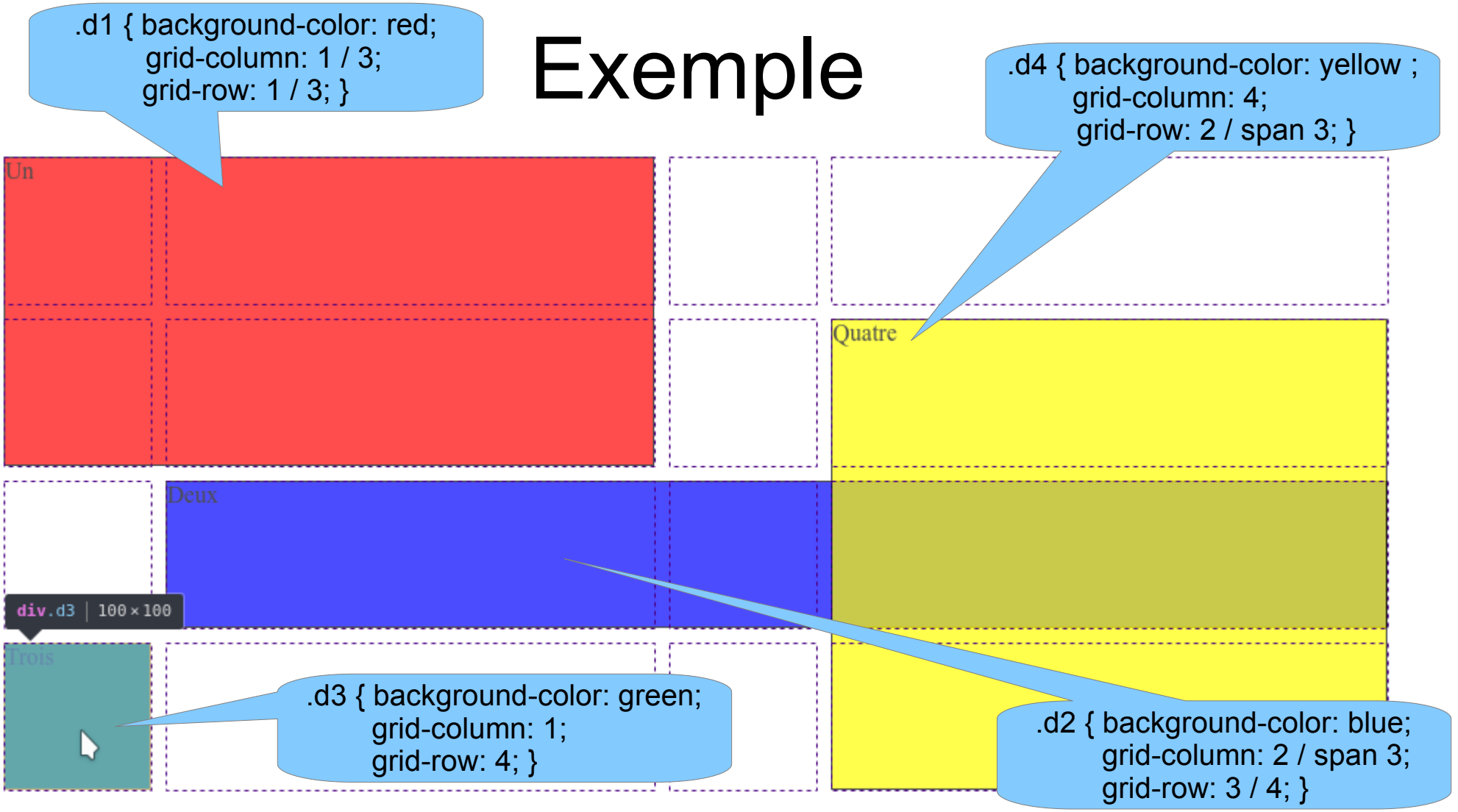
Disposition des items

- Exemples
  - `grid-column-start: 1;`
  - `grid-column-end: span 3;`
  - `grid-row-start: 2;`
  - `grid-row-end: r1;`
- **Raccourcis** : `debut / fin`
  - `grid-column: 1 / 2;`
  - `grid-row: r1 / span 4;`

Entre colonne 1 et colonne 2 (2 exclue, donc taille horiz = 1).

Commence ligne « r1 » et s'étend sur 4 lignes

Exemple




```
<div class="wrapper">
  <div class="d1">Un</div>
  <div class="d2">Deux</div>
  <div class="d3">Trois</div>
  <div class="d4">Quatre</div>
</div>
```

```
.wrapper {
  display: grid;
  grid-template-columns: repeat(2, 100px auto);
  grid-template-rows: repeat(4, 100px);
  gap: 10px;
}
.wrapper > div { opacity: .7; border: solid 1px black; }
```

Définition des dimensions de grilles : compléments

- Des syntaxes de plus
  - `grid-template-rows: minmax(10em, 2fr) 50px;`
  - `grid-template-columns: repeat(auto-fill, minmax(100px, 1fr));`
  - `grid-template-columns: repeat(auto-fit, minmax(100px, 1fr));`

1ère ligne : au min 10em, et jusqu'à 2 fr

Définit autant de colonnes que possible (larg. min. de 100px), qu'il y ait des items ou non pour les remplir

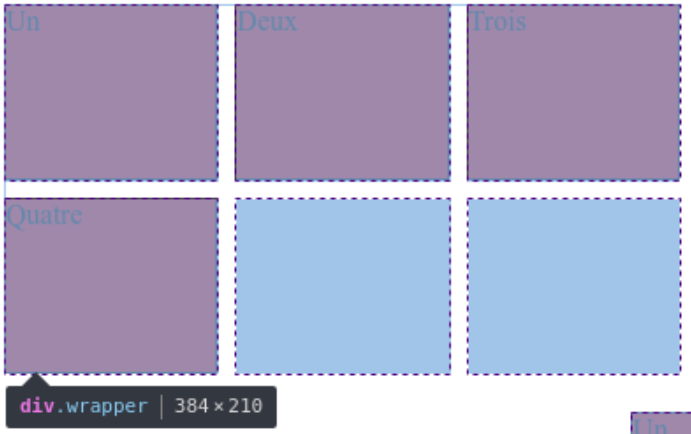
Définit un nombre de colonnes **automatique et dynamique**

Ajuste le nombre et la largeur des colonnes (larg. min. de 100px) en fonction des items présents

auto-fill

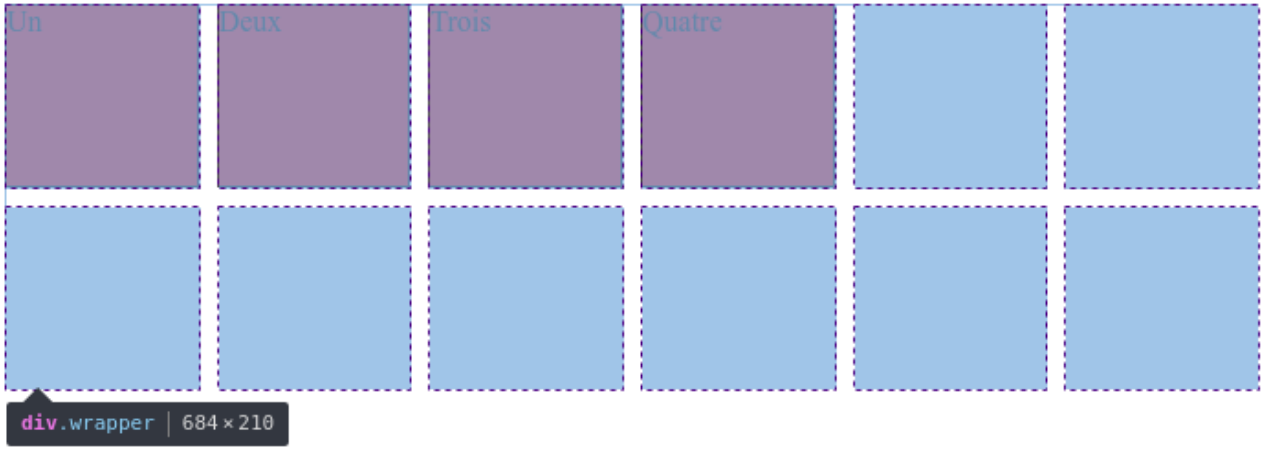
```
<div class="wrapper">
   <div class="d1">Un</div>
   <div class="d2">Deux</div>
   <div class="d3">Trois</div>
   <div class="d4">Quatre</div>
 </div>
```

```
.wrapper {
  display: grid;
  grid-template-columns: repeat(auto-fill, minmax(100px, 1fr));
  grid-template-rows: repeat(2, 100px);
  gap: 10px;
}
```



Largeur : 400px

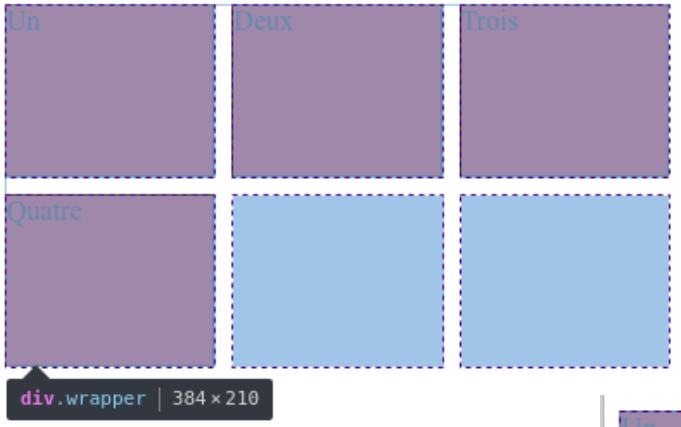
Largeur : 700px



auto-fit

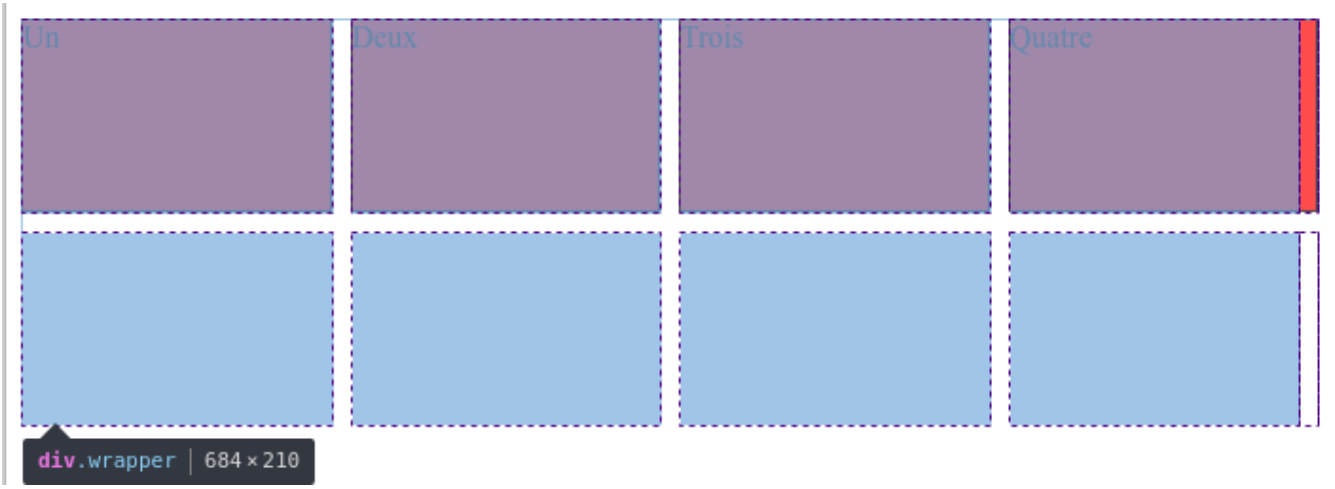
```
<div class="wrapper">
   <div class="d1">Un</div>
   <div class="d2">Deux</div>
   <div class="d3">Trois</div>
   <div class="d4">Quatre</div>
 </div>
```

```
.wrapper {
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(100px, 1fr));
  grid-template-rows: repeat(2, 100px);
  gap: 10px;
}
```



Largeur : 400px

Largeur : 700px



Autres propriétés **grille**

- `gap` (anciennement `grid-gap`) : espacement entre cellules
- `justify-items` : alignement horizontal des items sur la grille
  - utile si les items ont des dimensions < à la cellule
  - un peu le même principe qu'avec flex
  - `start, end, center, stretch`
- `align-items` : alignement vertical des items sur la grille
  - `start, end, center, stretch`

Propriétés **items**

- Alignements internes (aux items)
  - `justify-self` : selon l'axe principal (horizontal)
  - `align-self` : selon l'axe secondaire (vertical)

Exercice

- Faire les exercices du TD 7