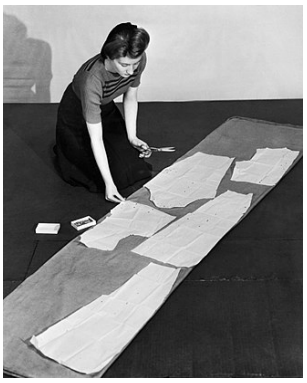
Template

L'idée d'utiliser un modèle (template) est une idée très générale.

En couture, le terme de patron (ou modèle) est largement utilisé.

> "Un **patron** … permet de concevoir un vêtement avant sa fabrication en couture"[1]



⏩ La notion de patron (template) est également répandue en informatique[2].

Le mot template existe explicitement dans un langage comme HTML[3].

Plus d'info : la balise <template>[4] !

Nous trouvons également en JS le terme de "template string". Voici deux exemples de code utilisant le template string noté ``.

1. Affichage dans la console
2. Modification du DOM

Nous allons analyser un cas concret d'utilisation d'un template dans la conception d'un calendrier.

Mais, revenons maintenant à la partie calcul qui génère un tableau de valeurs.

Partie Calcul

Voici la partie calcul[5] qui génère un tableau de valeur. Les valeurs correspondent au mois d'octobre 2020.

```
1. function calendarT(month = 9, year = 2020) {
2.
```

---

[1] wikipedia
[2] design pattern (↪ lien)
[3] la balise template
[4] https://caniuse.com/?search=template
[5] _python

```
    const MONTH_Map = new Map([
        [0, 'Janvier'],
        [1, 'Février'],
        [2, 'Mars'],
        [3, 'Avril'],
        [4, 'Mai'],
        [5, 'Juin'],
        [6, 'Juillet'],
        [7, `Août`],
        [8, `Septembre`],
        [9, `Octobre`],
        [10, `Novembre`],
        [11, `Décembre`],
    ]),
        OFFSET_MONTH[6] = new Map([
            [1, 6],//lundi
            [2, 0],
            [3, 1],
            [4, 2],
            [5, 3],
            [6, 4],
            [0, 5] //dimanche
        ]);

    let firstOfMonth = new Date(year, month, 1),
        dayFirst = firstOfMonth.getDay(),// Sunday - Saturday : 0 - 6
        dec = OFFSET_MONTH.get(dayFirst),
        cal = [];

    for (let i = 0; i < 42; i++) {
        let d = new Date(year, month, i - dec);
        cal.push(d);
    }
    return cal;
}

calendarT();
```

Remarques :

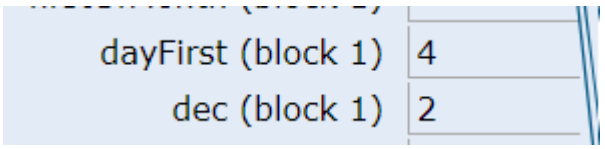
Le premier jour du mois d'octobre 2020 est un jeudi, dayfirst = 4.

Le décalage pour l'affichage est de 2.

lig. 33 : Il y aura trois jours à afficher du mois précédent.

---

[6] OFFSET permet de décaler en fonction du premier jour du mois le début du calendrier. Si le premier jour est un Lundi on affichera une semaine du mois précédent.



Nous voulons afficher les valeurs de ce tableau à l'aide d'un template. Un template définit un motif de base que l'on doit dupliquer un grand nombre de fois.

Template de base

Pour découvrir la mécanique des templates, nous allons prendre un exemple simple.

L'idée est d'afficher à l'aide d'une grille (7,6) les valeurs d'une liste. Le template représente le motif de base à dupliquer ; c'est-à-dire un élément de la liste.

L'élément de la liste de base en HTML est la balise <li>.

Définissons ainsi le template :

```
1. const defaultTemplate = `<li>{{data}}</li>`
```

La valeur data dépendra de l'indice du tableau à afficher. Nous allons comprendre le rôle des {{}} dans très peu de temps.

Pour remplir notre liste, il nous faut dupliquer ce template autant de fois qu'il y a de valeurs.

Nous pouvons itérer sur les valeurs du tableau comme suit

```
1. function show(calT) {
2.     let view = '';
3.
4.     for (let data of calT) {
5.
6.         let template = defaultTemplate;
7.
8.         template = template.replace('{{data}}', data.getDate());
9.
10.        view = view + template
11.    }
12.
13.    return view
14. }
```

En lig.8, nous remplaçons {{data}} à l'aide de la valeur courante.

La méthode replace[7] est extrêmement puissante.

Lig.10, la concaténation accumule les valeurs de la liste.

Nous transformons notre tableau de 42 valeurs en un string de 42 li :

```
'<li>28</li><li>29</li><li>30</li><li>1</li><li>2</li><li>3</li><li>4</li><li>5</li><li>6</li><li>7</li><li>8</li><li>9</li><li>10</li><li>11</li><li>12</li><li>13</li><li>14</li><li>15</li><li>16</li><li>17</li><li>18</li><li>19</li><li>20</li><li>21</li><li>22</li><li>23</li><li>24</li><li>25</li><li>26</li><li>27</li><li>28</li><li>29</li><li>30</li><li>31</li><li>1</li><li>2</li><li>3</li><li>4</li><li>5</li><li>6</li><li>7</li><li>8</li>'
```

Il ne reste plus qu'à l'ajouter au DOM. Attention la valeur à ajouter n'est pas un élément. Nous utilisons la méthode inserAdjacentHTML[8].

```
ul.insertAdjacentHTML('afterBegin', show(calendarT()));
```

code

Projet

Nous voudrions mettre en avant le jour courant !

28	29	30	1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	1
2	3	4	5	6	7	8

Aide : l'inspection du code donne

28	29	30	1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18

```
<li id="1601848800000">5</li>
<li id="1601935200000">6</li>
<li id="1602021600000">7</li>
<li id="1602108000000" class="today">8</li>
<li id="1602194400000">9</li>
<li id="1602280800000">10</li>
<li id="1602367200000">11</li>
<li id="1602453600000">12</li>
```

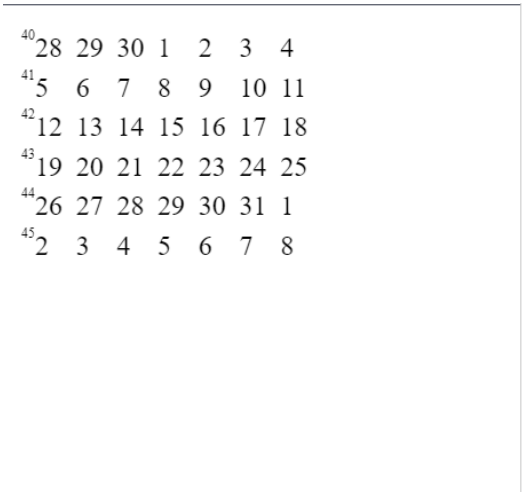
[7] replace mdn
[8] insertAdjacent

Template calendrier p.5

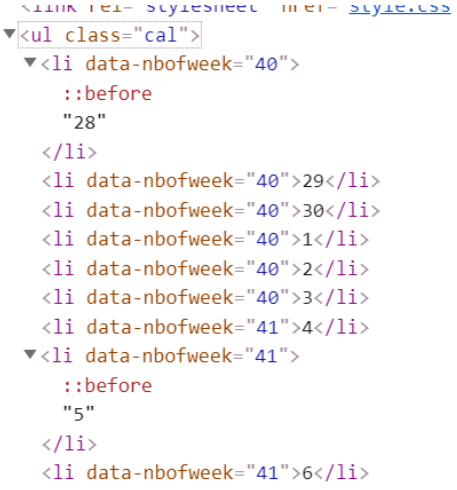
40 28

Projet 41 5

Nous voudrions ajouter la semaine sur le lundi



40 28 29 30 1 2 3 4
41 5 6 7 8 9 10 11
42 12 13 14 15 16 17 18
43 19 20 21 22 23 24 25
44 26 27 28 29 30 31 1
45 2 3 4 5 6 7 8



```
<link rel="stylesheet" href="style.css">
▼<ul class="cal">
  ▼<li data-nbofweek="40">
      ::before
      "28"
    </li>
    <li data-nbofweek="40">29</li>
    <li data-nbofweek="40">30</li>
    <li data-nbofweek="40">1</li>
    <li data-nbofweek="40">2</li>
    <li data-nbofweek="40">3</li>
    <li data-nbofweek="41">4</li>
  ▼<li data-nbofweek="41">
      ::before
      "5"
    </li>
    <li data-nbofweek="41">6</li>
```