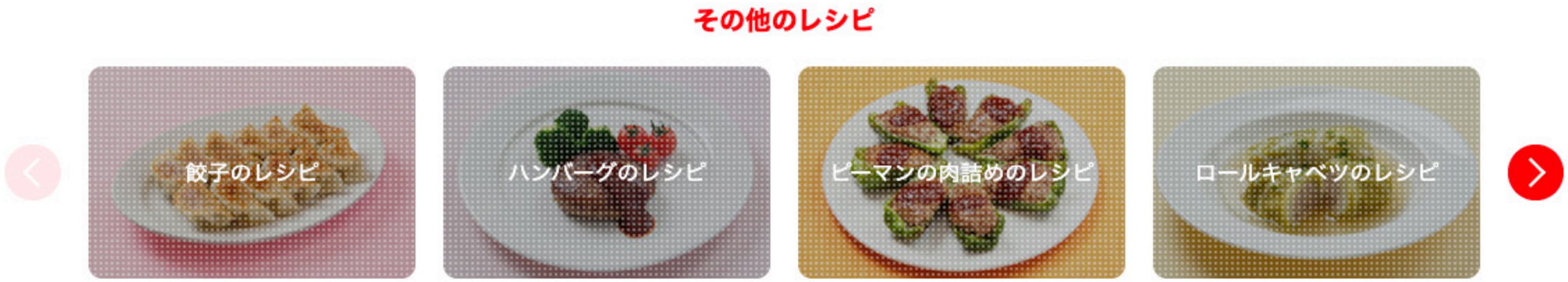
Web プログラミング

第十二回：JavaScript(2)

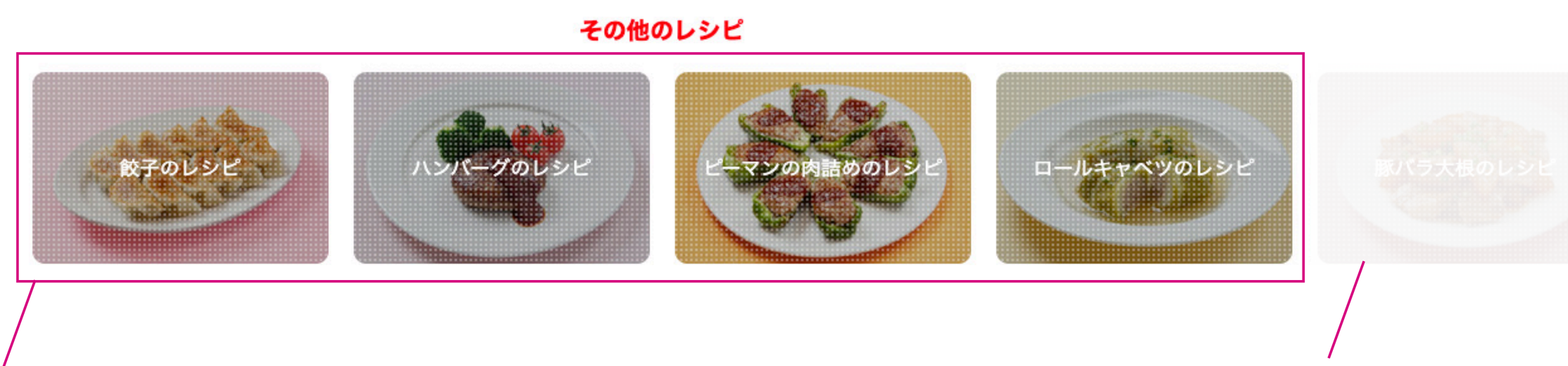
JS(2) - カルーセルの実装

「その他のレシピ」の箇所を、
左右の矢印クリックでページ送りするような形に書き換えましょう。
このようなレイアウトを、俗に「カルーセル」と呼びます。



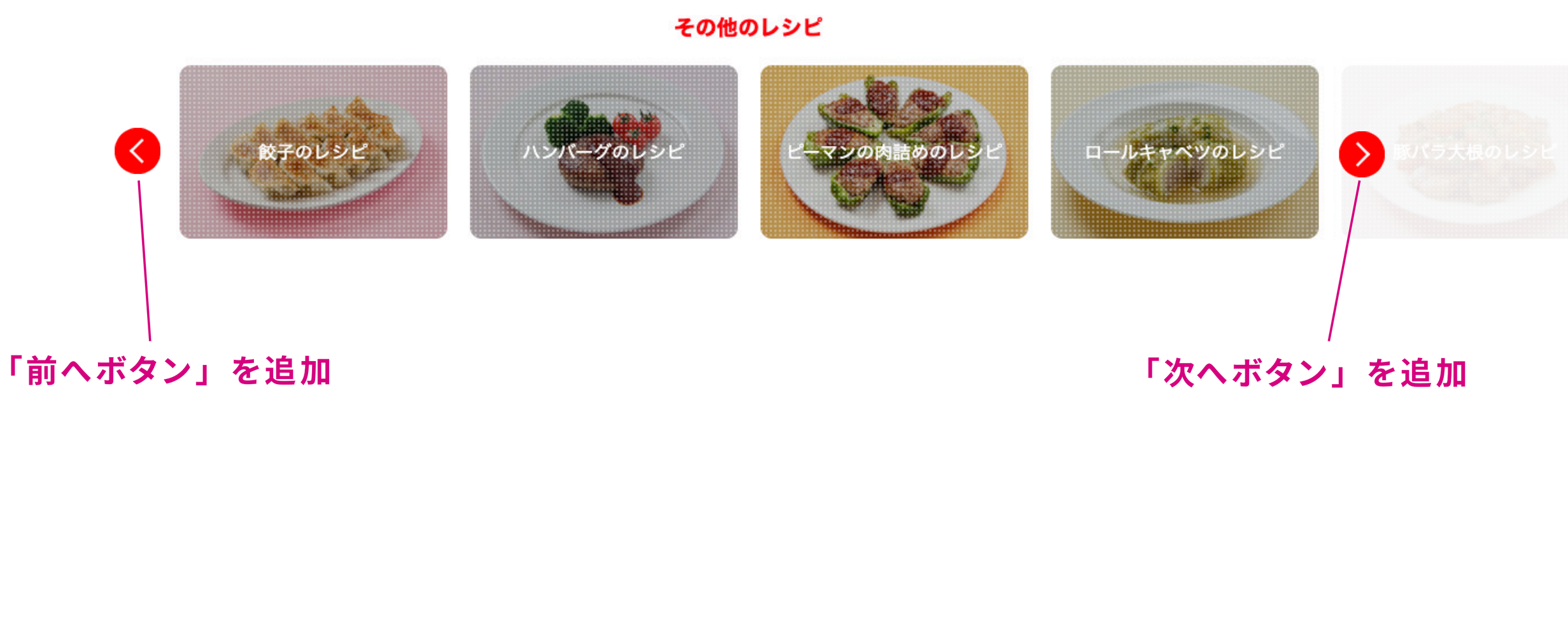
JS(2) - 仕組みの考え方と設計

どのようなプログラムを書けば、カルーセルレイアウトが作れるかを考えてみます。
まず、ボックスを全て横一列にならべます。
見える範囲からはみ出している部分は表示されないような形にします。



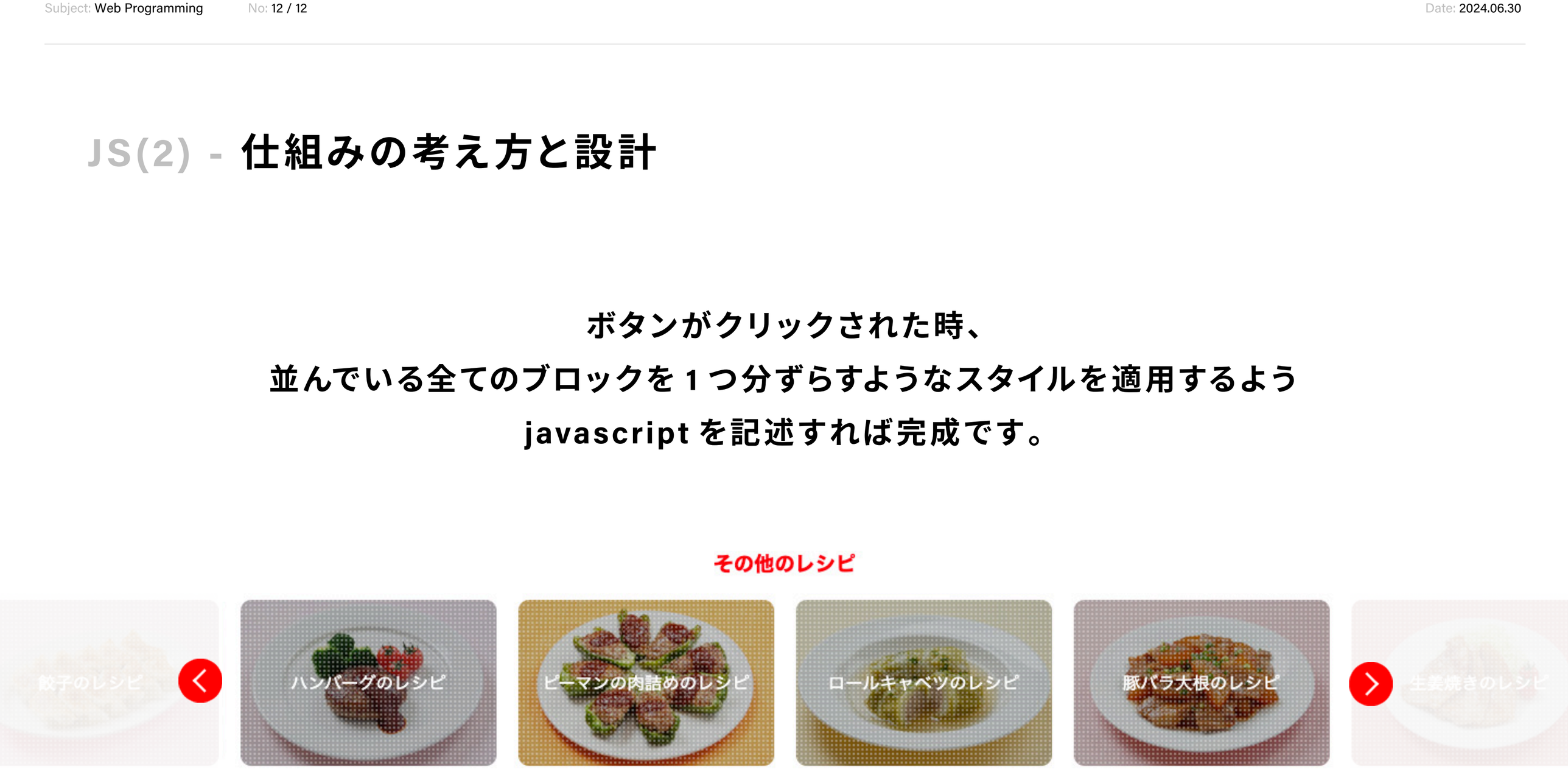
JS(2) - 仕組みの考え方と設計

並んでいるボックスの左右に、
「次へボタン」と「前へボタン」を追加します。



JS(2) - 仕組みの考え方と設計

ボタンがクリックされた時、
並んでいる全てのブロックを 1 つ分ずらすようなスタイルを適用するよう
javascript を記述すれば完成です。

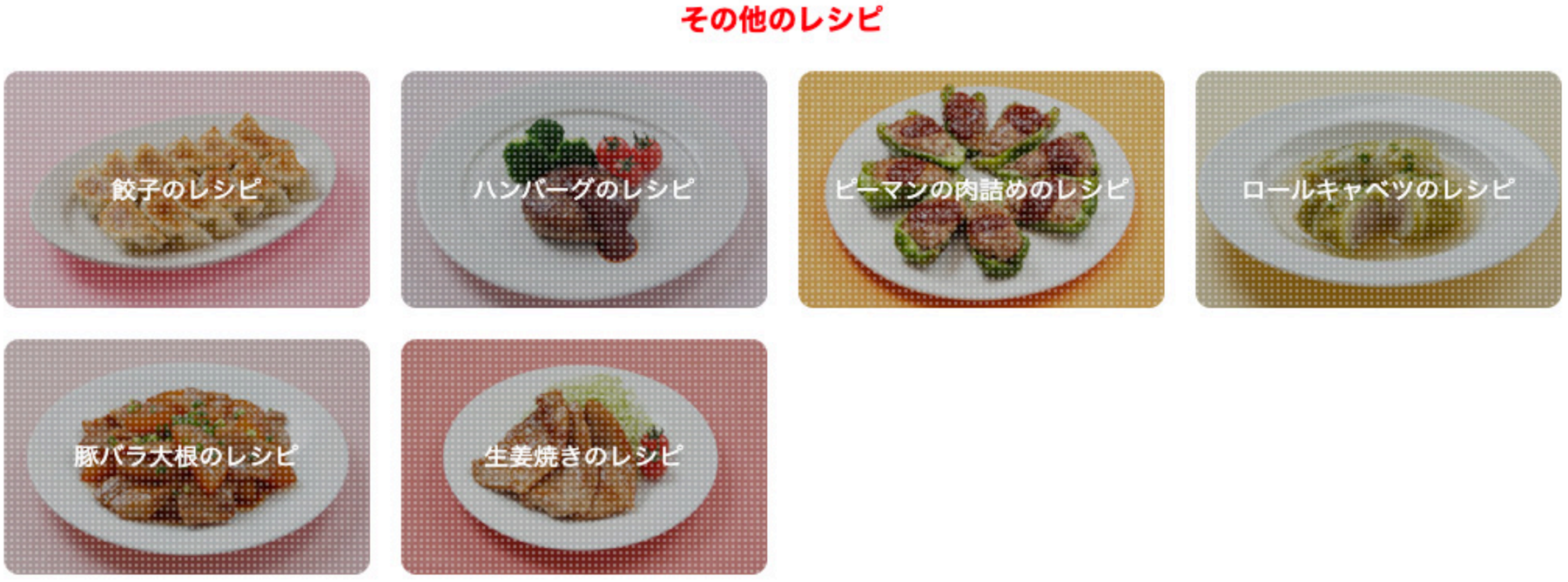


ボタンがクリックされたら、
並んでいる各ブロックを全て一つ分ずらす

JS(2) - レイアウトの調整

javascript を実装する前の下準備として、
HTML と CSS を調整してレイアウトを作成します。

現状、各リンクブロックは flex-flow: row wrap の
フレックスボックスの中にいるので、横に並び、端で折り返しています。

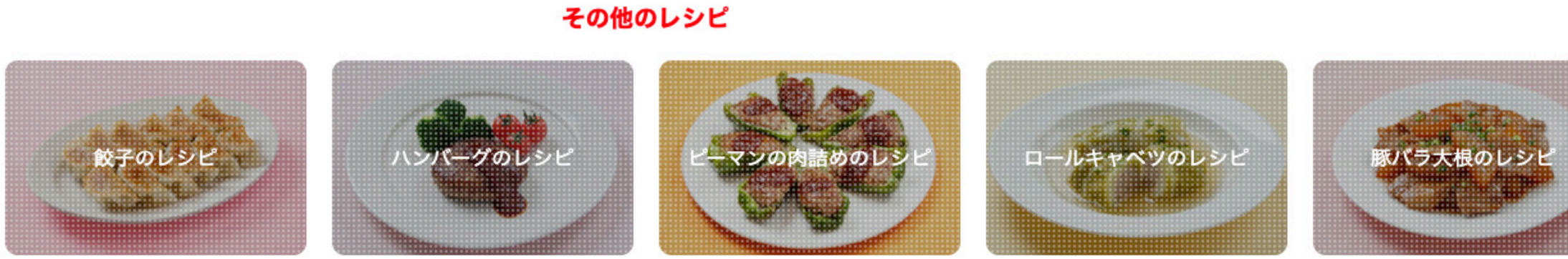


JS(2) - レイアウトの調整

各リンクブロックを囲んでいる親ボックスを

flex-flow: row nowrap

に修正し、端で折り返さず、横にはみ出すようにしてみましょう。



JS(2) - レイアウトの調整

親ボックスに

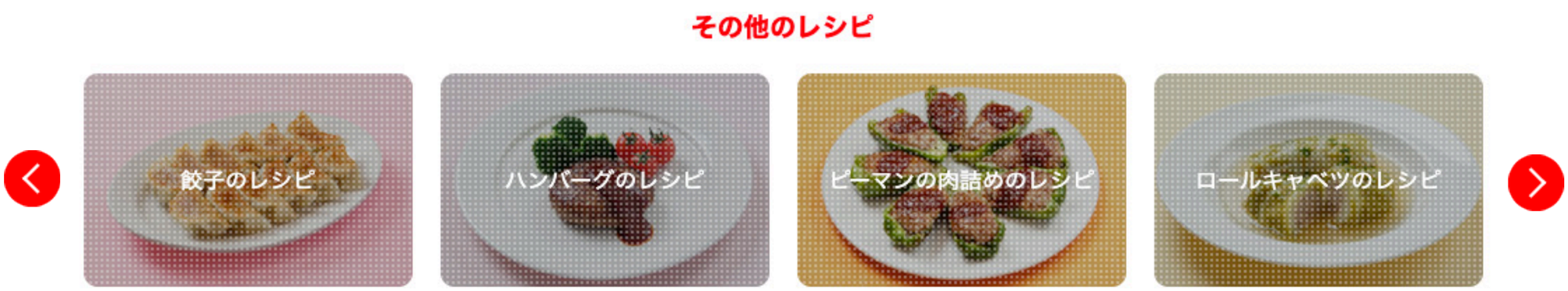
overflow:hidden

を設定すると、領域からはみ出したものは表示しないようにできます。

JS(2) - レイアウトの調整

HTML を編集し、「次へボタン」と「前へボタン」となる
<button> タグを追加します。
<button> タグにスタイルを適用し、ブロックの左右にくるよう調整します。
これで準備は完了です。

```
<button class="links_prev"><img src="images/prev.png" alt=""></button>
<button class="links_next"><img src="images/next.png" alt=""></button>
```



JS(2) - スクリプトの実装

document.querySelector を使って、

2 つのボタンを javascript の変数に読み込みましょう

```
var nextButton = document.querySelector(".links_next");
var prevButton = document.querySelector(".links_prev");
```

JS(2) - スクリプトの実装

addEventListener を使って、

ボタンをクリックした時に関数が実行されるようにしましょう

```
var nextButton = document.querySelector(".links_next");
var prevButton = document.querySelector(".links_prev");

function moveToNext(){
    console.log(" 次へボタンがクリックされた ");
}
function moveToPrev(){
    console.log(" 次へボタンがクリックされた ");
}
nextButton.addEventListener("click", moveToNext);
prevButton.addEventListener("click", moveToPrev);
```

JS(2) - スクリプトの実装

現在位置を覚えておくための変数 position を宣言し、
次へボタンが押されたら +1、前へボタンが押されたら -1 するようにしましょう。

```
var nextButton = document.querySelector(".links_next");
var prevButton = document.querySelector(".links_prev");
var position = 0;

function moveToNext(){
    position++;
    console.log(position);
}
function moveToPrev(){
    position--;
    console.log(position);
}
nextButton.addEventListener("click", moveToNext);
prevButton.addEventListener("click", moveToPrev);
```

JS(2) - スクリプトの実装

document.querySelectorAll を使って、
全てのボタンブロック要素を配列の形で取得し、変数に保存します。

```
var nextButton = document.querySelector(".links_next");
var prevButton = document.querySelector(".links_prev");
var position = 0;
var linksItem = document.querySelectorAll(".links_item");

function moveToNext(){
    position++;
    console.log(position);
}
function moveToPrev(){
    position--;
    console.log(position);
}
nextButton.addEventListener("click", moveToNext);
prevButton.addEventListener("click", moveToPrev);
```

JS(2) - スクリプトの実装

ボタンがクリックされ position が増えたり減ったりしたら、
リンクブロック全てに、position に応じた「ずれる」css を適用するようにしましょう

```
... 省略 ...

function moveToNext(){
    position++;
    for(var i=0; i<linksItem.length; i++){
        linksItem[i].style.left = (position * 255 * -1)+"px";
    }
}
function moveToPrev(){
    position--;
    for(var i=0; i<linksItem.length; i++){
        linksItem[i].style.left = (position * 255 * -1)+"px";
    }
}

... 省略 ...
```

JS(2) - スクリプトの実装

このままでは、クリックしただけ無限に左右に移動し続けることができます。
移動の上限を設定するために、「何個ずれることができるか」の上限を計算します。

```
... 省略 ...

var maxMove = linksItem.length-4;

function moveToNext(){
    position++;
    for(var i=0; i<linksItem.length; i++){
        linksItem[i].style.left = (position * 255 * -1)+"px";
    }
}
function moveToPrev(){
    position--;
    for(var i=0; i<linksItem.length; i++){
        linksItem[i].style.left = (position * 255 * -1)+"px";
    }
}

... 省略 ...
```

JS(2) - スクリプトの実装

次へを押した時には、position は上限以上に増えないよう、
逆に前へを押した時は、0 以下に減らないよう if 文で制御します。

```
... 省略 ...
function moveToNext(){
    if(position < maxMove){
        position++;
    }
    for(var i=0; i<linksItem.length; i++){
        linksItem[i].style.left = (position * 255 * -1)+"px";
    }
}
function moveToPrev(){
    if(position > 0){
        position--;
    }
    for(var i=0; i<linksItem.length; i++){
        linksItem[i].style.left = (position * 255 * -1)+"px";
    }
}
... 省略 ...
```

JS(2) - スクリプトの実装

「各ボタンブロックを左右にずらす css を適用する」処理は共通です。
冗長なので、1 つの関数として括り出してしまいましょう。

```
... 省略 ...
function moveToNext(){
    if(position < maxMove){
        position++;
    }
    move();
}
function moveToPrev(){
    if(position > 0){
        position--;
    }
    move();
}
function move(){
    for(var i=0; i<linksItem.length; i++){
        linksItem[i].style.left = (position * 255 * -1)+"px";
    }
}
... 省略 ...
```

JS(2) - スクリプトの実装

端まで移動したら、それ以上ボタンが押せないことを示すために見た目を変化させたい。
そのために、もし position が 0 か上限であったら、ボタンにクラスを追加しましょう。

```
... 省略 ...

function move(){
    for(var i=0; i<linksItem.length; i++){
        linksItem[i].style.left = (position * 255 * -1)+"px";
    }
    if(position==maxMove){
        nextButton.classList.add("disable");
    }else{
        nextButton.classList.remove("disable");
    }
    if(position==0){
        prevButton.classList.add("disable");
    }else{
        prevButton.classList.remove("disable");
    }
}

... 省略 ...
```

JS(2) - スクリプトの実装

完成した javascript のソースコード

```
var nextButton = document.querySelector(".links__next");
var prevButton = document.querySelector(".links__prev");
var linksItem = document.querySelectorAll(".links_item");
var position = 0;
var maxMove = linksItem.length-4;

function moveToNext(){
    if(position < maxMove){
        position++;
    }
    move();
}
function moveToPrev(){
    if(position > 0){
        position--;
    }
    move();
}
function move(){
    for(var i=0; i<linksItem.length; i++){
        linksItem[i].style.left = (position * 255 * -1)+"px";
    }
    if(position==maxMove){
        nextButton.classList.add("disable");
    }else{
        nextButton.classList.remove("disable");
    }
    if(position==0){
        prevButton.classList.add("disable");
    }else{
        prevButton.classList.remove("disable");
    }
}

nextButton.addEventListener("click", moveToNext);
prevButton.addEventListener("click", moveToPrev);
```