スライドギャラリー

HTML

1. スライドギャラリー全体を表示する要素は div#slideGalley です。
2. li 要素にスライドする画像を入れています。
3. prev ボタンと next ボタンは ul 要素のすぐ後ろに#nav として設置
4. ボタンの下にページングの小さい●を表示する ul を置きます。li 要素はスクリプトで作成します。

CSS

1. ul#slide は横幅を 10000px も確保しますので、div#slideGalley をはみ出す構造になります。そこで div#slideGalley には width:650px として、さらにはみ出した部分を消すため overflow:hidden;を設定します。また、next ボタンなどを絶対配置しますので、position: relative を設置します。
2. ul#slide の横幅は li をフロートし、また li の数（画像の数）が増えることも考慮して width:10000px とします。
3. 付属のパーツとして、nav_prev ボタンと nav_next ボタン並びに現在の画像の順番を示す paging は絶対配置で設置します。
4. #paging li 要素は display: inline-block で横並びにし、高さと幅を指定します。
5. #paging li 要素の width と height は背景画像の●の大きさにあわせます。
6. #paging li 要素でアクティブになった li は背景画像を変更したいので.active を付けて変更します。

スクリプト

PAGE LOAD 後

1. まずは、paging の丸印を作成。#slide にeach() を使用することでスライドの枚数だけ paging の丸印を作成する。

```
<ul id="paging">

<li data-img="./images/photo01.jpg" class="active"></li>

<li data-img="./images/photo02.jpg"></li>

<li data-img="./images/photo03.jpg"></li>

<li data-img="./images/photo04.jpg"></li>

</ul>
```

2. また最初のli をclass="active" にする。
3. <li data-img="./images/photo01.jpg" class="active">
4. data-img は HTML5 のカスタムデータ属性です。独自の属性で値を一時的に保管する場合に好都合です。ブラウザで完成例を表示して要素検証で li の属性を確認してみましょう。

NEXT ボタンクリック

1. ″margin-left″ : -1*$(″#slide li″).width()

   画像１枚の幅分だけマイナスマージンをかける（左に引っ張る）。$(″#slide li″).width()とすることで画像の幅をここで取得しているので、画像の幅が変更されてもスクリプトの変更は必要ない。

2. 画像1枚の幅を取得できたら、コールバック関数で左マージンを０にする。と同時に最初の画像を最後に持っていく。この動作は.css()と.append()をチェーンメソッドで実行

3. クリックするごとに最初のli要素を最後に入れ替えるため、表示画像は常に1 枚目のliになる（li:first-child)

4. removeClass(″active″) でactive のクラス名を消す

5. $(″#paging li[data-img='″+$(″#slide li:first-child img″).attr(″src″)+″']″) とはdata-img=が現在の最初の画像が入っているli のアドレスになります。これと同じ値のdata-imgを探すことになる。

   つまり１で作成したpaging のliのdata-imgの値が同名のものに. activeを付けることになる。

PREV ボタンクリック

1. .css(″margin-left″,-1*$(″#slide li″).width())

   こちらはcssメソッドで画像幅分だけマイナスマージンをかける。cssメソッドを使用しているので一瞬で画像幅分のネガティブマージンがかかります。

2. 次にチェーンメソッドで最後のliの画像が一番先頭にきます。

3. １，２は一瞬の間に起こります。その後animate()で″margin-left″ : 0にするため左に隠れていた最後のliから移動してきていた画像がゆっくり右に移動しながら現れます。

4. コールバック関数はこちらは#paging liへのクラス名activeの付け替えになります。内容はnextと同様です。すでにliの移動は終えていますので、nextの時のようにコールバック関数内で処理する必要はありません。

自動スライドの実行

```
var timerId = setInterval(function(){
        $("#nav .next").click();
},5000);
```

自動スライドの実行はカルーセルの時と同じです。

setInterval("関数名" , 起動するミリ秒);

自動スライドの制御

```
$("#slideGalley").hover(function(){
        $("#nav").show();
        clearInterval(timerId);
},function(){
        $("#nav").hide()
        timerId = setInterval(function(){
                $("#nav .next").click();
                },5000);
});
```

hover(over, out)

マウスホバーの動きをシミュレートします。
マウスカーソルが要素の上に乗った時に、第一引数に渡した関数を実行します。マウスが要素から外れた時には第二引数が実行されます。

1. $("#slideGalley").hover で#slideGalley 領域にマウスが入るとボタンを表示します。
2. setInterval を止めます。つまり手動でボタン操作をしてスライドを操作するためにボタンを表示して、自動スライドを止めています。
3. 次の function の内容はマウスが外れたときの動作の命令です。
4. マウスが外れたらボタンを非表示にして、setInterval を開始しています。