location.href（リンク）

ここから、リンク関係の JavaScript について解説していきます。 まず始めは、普通にリンクを貼る方法です。 <a href="">と動作は同じです。 FORM のボタンにリンク指定したりする場合に使います。

location.href の記述

ではボタンにリンクを指定してみたいと思います。 HTML の body 内に以下のように記入してみて下さい。

```
<script>

function jump(){
  if (confirm("トップページに戻りますか？")==true)
    //OK なら TOP ページにジャンプさせる
    location.href = "http://pori2.net/";
}

</script>


<form>
<input type="button" name="link" value="TOP へ" onclick="jump()">
</form>
```

上記のサンプルは、関数 jump()を作成しています。 confirm でリンク先にジャンプするか確認するダイアログを表示します。

確認ダイアログでＯＫが押された場合、location.href=で指定した URL にジャンプします。

URL は文字列なので、クォーテーションで囲む必要があります。

location.href="URL"

指定した URL へジャンプする

最後に FORM 内にボタンを配置し、 onclick イベントで関数 lump()を呼び出します。

フレームの場合

フレームの場合は、location.href を使ってフレームを解除することができます。 以下のように先頭に文字を付け足します。

top.location.href="URL"

全フレームを解除します

parent.location.href="URL"

親フレームを解除します

HTML の target 属性に指定する値を思い浮かべれば、簡単だと思います。 このように location.href は HTML の A タグと同様の働きをさせることができます。
location.href は単独で用いることもあれば、 いろいろな処理と組み合わせて用いることもできます。 例えばクイズを作ったとしましょう。 全問正解するとパスワードが載せられた特定のページにジャンプさせたりすることができます。 このように使い道は色々とあるので、しっかり覚えて下さい。

セレクトボックスでナビゲーション

前回 location.href でリンクを貼る方法について学びました。 ではそれを応用して、セレクトボックスによるナビゲーションを作ってみましょう。 セレクトボックスによるナビは時々見かけますよね。

↑こんなタイプのナビです。 目的のものを選択すると、そのページに飛びます（上記のセレクトボックスにはリンクを張っていません)。

セレクトボックスでリンク先を指定する

では実際にソースの書き方を解説します。 以下のような関数とフォームを、body 内の表示したい部分に記入しましょう。 勿論関数類はヘッダーでも構いません。

```
<script>

//セレクトボックスに対応するリンク先を配列に入れる
var jpURL = new Array(
"http://pori2.net/",
"http://pori2.net/js/index.html",
"http://pori2.net/saku2/index.html"
);

//リンク先へジャンプさせる関数
function selectNavi(){
  var num;

  //何番目の option が選択されたか調べる
  num = document.navi.contents.selectedIndex;
```

```
 //該当するリンク先へジャンプさせる
 location.href = jpURL[num];
}

</script>


<form name="navi">
<select name="contents" onchange="selectNavi()">
 <option>HOME</option>
 <option>JavaScript 入門</option>
 <option>さくさくＨＰ作り</option>
</select>
</form>
```

ではまず form から解説していきます。 JavaScript と form を連動させる場合は、 form タグと select タグに name 属性を指定しなければいけません。上記では form タグに「navi」, select タグに「contents」という名前を付けました（名前は何でも構いません)。 そして選択項目（option タグ）を３つ設けています。 選択項目が変更になった時 onchange イベントが発生するので、 そこで関数 selectNavi()を呼び出します。

次に、JavaScirpt 部分を見ていきます。 まずは配列に、ジャンプ先の URL を指定していきます。配列 jpURL に、option タグの上から順に対応する URL を記入していって下さい。

次に、リンク先へジャンプさせる関数 selectNavi()の宣言です。 最初に変数 num を宣言しました。 これは何番目の項目が選択されたかを入れる変数です。

次にセレクトボックスの何番目の項目が選択されたかを調べます。document の後に、form と select ボックスの name 属性値が続きます。 最後に selectedIndex を記入します。これが選択項目の番号を取得するものです。 I が大文字ですから気をつけましょう。

document.form 名.select 名.selectedIndex
セレクトボックスの選択番号を取得する

最後は location.href でジャンプさせます。 ジャンプ先に jpURL[num]を指定することによって、 セレクトボックスの選択項目番号が配列の何番目の URL に該当するかを決定する仕組みです。

一番目の項目を選択した場合

上記スクリプトでは一つ問題があります。 実は HOME を選んだときに、ジャンプしません。 これは選択項目が変更されていないので（最初から HOME が選択された状態だから)、onchange イベントが発生しないからです。 HOME 以外のページから HOME に戻ろうと

しても、戻ることができません。
ということでこの問題を解決するために、 option タグの 1 番目は「コンテンツ一覧」とかにし、 HOME を 2 番目の項目にすると良いでしょう。 以下に修正したスクリプトを載せておきます（赤文字が修正部分）。

```
<script>

var jpURL = new Array(
"",  //ダミーの項目を一つ追加します
"http://pori2.net/",
"http://pori2.net/js/index.html",
"http://pori2.net/saku2/index.html"
);

function selectNavi(){
  var num;
  num = document.navi.contents.selectedIndex;

  //最初の項目以外が選択された時にジャンプする
  if ( num != 0 ) location.href = jpURL[num];
}

</script>

<form name="navi">
<select name="contents" onchange="selectNavi()">
  <option>コンテンツ一覧</option>
  <option>HOME</option>
  <option>JavaScript 入門</option>
  <option>さくさくＨＰ作り</option>
</select>
</form>
```

さて、最初の項目を選んでも onchange イベントが発生しないのであれば、上記の if 文は必要無いと思うかもしれません。 しかし別のページに飛んで、ブラウザの戻るボタンで戻ってみるとどうでしょうか。 セレクトボックスの選択項目は、先に選択したものになったままです。 このまま「コンテンツ一覧」を選択すると onchange イベントが発生してしまう訳です。 このような理由で、最初の項目が選択された場合はジャンプしないようにして

おきました。

ナビゲーションは外部ファイルで
こうしたナビゲーションを全てのページに置いておくと親切ですが、 全てのページにform,select,option を記入するのも面倒です。 option の数が上記のように３つということは無く、普通はもっと多いからです。 コンテンツを追加した場合は全てのページを修正することになり大変です。
そんな時は、全てを外部ファイルで操作しましょう。 フォーム関連のタグもdocument.write()で書き出してしまえばいいわけです。外部ファイルを１つ作っておけば、各ページから読み込んですぐにナビを設置できます。 修正も外部ファイルのみ変えればＯＫです。
外部ファイルの例（別窓） サンプル（別窓）
あとは各ページから外部ファイルにリンクし、目的の場所で関数 formWrite()を呼び出せばＯＫですね。 因みに formWrite()内で document.write()が頻出していますが、 配列と for 文を使えばもう少しすっきりさせることが可能です。

戻る・進む・更新ボタンを作る
ここでは、ブラウザの戻る・進む・更新といったボタンと同じ機能を持ったボタンを、JavaScript で作ってみたいと思います。 特に「戻る」ボタンなどは、 色々な所からリンクされているページに使うと威力を発揮します。
「更新」ボタンはリンクなどで出てきた location を使いますが、「戻る」と「進む」は history というものを使います。 英語で「履歴」という意味ですね。

戻る・進む・更新ボタンの作り方
戻るや進むボタンのソースは１行なので、HTML のボタン内に直接書きこむことができます。 が、ページの横幅の関係上、一応関数形式にして書いています。 外部ファイルに関数を記入しておけば、 全てのページで使えるので便利かもしれません。 まあそんなに変わらないでしょうけど。

```
<script>

//戻る
function modoru(){
  history.back();
}

//進む
```

```
function susumu(){
  history.forward();
}

//更新
function koshin(){
  location.reload();
}

</script>

<form>
<input type="button"  value="戻る" onclick="modoru()">
<input type="button"  value="進む" onclick="susumu()">
<input type="button"  value="更新" onclick="koshin()">
</form>
```

上記のように、history.back()が「戻る」， history.forward()が「進む」， location.relord()が「更新」を意味します。 それぞれ関数にしています。 それをフォームのボタン内にある onclick イベントで呼び出してやればＯＫです。

history.back()
一つ前のページに戻ります（ブラウザの戻るボタンと同じ）
history.forward()
一つ先にページに進みます（ブラウザの進むボタンと同じ）
location.relord()
ページを更新します（ブラウザの更新ボタンと同じ）

当然ながら、戻るページが無い（進むページが無い）という時には移動できません。 ということで、今回サンプルページは用意していません。 新しいページを開いた場合、戻るも進むも使えませんのでね。 このページ内のボタンでテストしてみて下さい。

複数ページ飛び越えて移動する

history には、複数のページを飛び越えて戻ったり進んだりする機能もあります。 何に使えるかは分かりませんが、知識として取り入れておくと良いでしょう。

```
<script>

//履歴を指定数移動する
function rireki(num){
```

```
        history.go(num);
}

</script>

<form>
<input type="button"  value="２戻る" onclick="rireki(-2)">
<input type="button"  value="３進む" onclick="rireki(3)">
</form>
```

履歴を指定数だけ移動するには、history.go()を使います。 カッコ内には移動ページ数を入れます。負の数を入れると戻り、 正の数を入れると進みます。 上記の場合は引数 num を指定し、関数呼び出し時に num に移動数を代入しています。 それが history.go のカッコ内に代入される仕組みです。

history.go()

カッコ内に指定された数だけページを移動します。マイナスを指定すると戻ります。

戻れないページを作る

今回は、ブラウザの「戻るボタン」 を押しても戻ることができないようにしてみたいと思います。 この機能は検索エンジンなどで望まないページにアクセスされた時に、 強制的に TOP ページへジャンプさせたりする時に使います。

今のページに戻れなくするには、ブラウザの履歴に URL を登録させないようにします。 A タグや location.href を使うと現在のページが履歴に残るので、 location.replace()というものを使います。

location.replace()でリンクさせる

では、location.replace()を使ってリンクボタンを作ってみましょう。 HTML の body 内に以下のように記入してみましょう。

```
<script>
function goTop(){
  location.replace("http://pori2.net/");
}
</script>

<form>
<input type="button" value="サンプル" onclick="goTop()">
</form>
```

上記のサンプルボタンを押すと、うちのサイトの TOP ページへ移動します。 そこでブラ

ウザの戻るボタンを押しても、このページに戻ることができません。 確認して下さい。戻れなくなりますが（汗）
location.replace()の場合は、後ろの括弧内にジャンプ先を記入します。 location.href は「=」で指定しますが、方法が違うので注意しましょう （私もよく間違えます）。
location.replace(URL)
履歴を残さずに、カッコ内に指定した URL に移動します

ページの強制移動
では検索エンジンで下層ページに来た場合に、TOP へ強制移動させる方法について掲載してみます。
この場合はまずどこからリンクしてきたかを調べて、 URL に「search」という文字が含まれていると TOP ページに強制移動させます。 戻るボタンを押されても履歴に URL は登録されていないので、 ジャンプ元のページではなくその前の検索結果ページに戻ります。
強制移動するページの body 内の一番上に、次のように書いておきます。

```
<script>
//変数 str に、直前のページ URL を入れる
var str = document.referrer;

//str 内に「search」という文字列があるか調べる
var sflg = str.indexOf( "search" , 0 );

//search 文字列がある場合に強制移動
if ( sflg != -1 )
  location.replace( "http://pori2.net/" );

</script>
```

では上から解説していきます。まず直前のページの URL を調べるには、document.referrer を使います。 URL を変数 str に代入しています。
document.referrer
直前のページ URL を取得します
次に、特定の文字が含まれているかどうか確かめるため indexOf()を使います。 調べたい文字列を前に書き、ピリオドで結んで indexOf を書きます。 カッコ内は最初に調べたい文字列を、 次に文字列の検索スタート位置を書きます。 スタート位置は通常 0 でＯＫです。
もしも検索文字列が見つかったなら、 変数 sflg に文字列の位置（前から何番目の文字か）が入ります。 見つからない場合は-1 が入ります。

文字列.indexOf(検索文字列 , 検索スタート位置)
文字を検索し、Hit すると何番目の文字かを取得します。
見つからない場合は-1 を返します。
最後に if 文で、sflg を調べます。 slfg が-1（search と言う文字が見つからなかった場合）はそのまま、 見つかった場合は location.replace()で強制移動します。 移動先はご自身のサイトの TOP ページ URL にして下さい。
まあこのようにして強制移動させられますが、 大抵の場合はお客さんを逃がすことになります（汗）。 ということで、あまり多用しない方が良いでしょう。
インラインフレームのリンク指定
インラインフレームに表示するサイト URL を変更する方法について見ていきましょう。
JavaScript を用いると、 インラインフレーム内の URL を自由に変更できます。 ただし他人のサイトを表示するのはご法度なので、 あくまで自分のサイト内のページに留めておきましょう。

リンク変更のサンプル
では HTML のヘッダーに、以下のように記入してみて下さい。

```
<script>

function ifr(jpURL){
  waku.location = jpURL;
}

</script>
```

次いで、body 内に以下のソースを書き込みましょう。

```
<a href="JavaScript:ifr('http://pori2.net/puzzle/')">
パズルの部屋へ
</a>
<br>

<iframe src="http://pori2.net/" name="waku"
 width="100%" height="80%"></iframe>
```

サンプルをご覧下さい。最初はうちのサイトの TOP ページが表示されていますが、 「パズルの部屋へ」というリンクを押すと、パズルの部屋が表示されますね。 インラインフレーム内のリンク先が変更されたからです。

リンク変更の仕組み

では上記のソースを詳しく見ていきましょう。
最初は関数 ifr の宣言をしています（関数名は iframe の略号のつもり）。 引数 jpURL を指定し、ここにリンク先を記入するようにしています。
関数内は１行だけです。最初の waku というのは、 インラインフレームの name 属性で指定した値です。 該当のインラインフレームのリンク先（location）に、jpURL を指定します。 location を location.href にしてもＯＫです。
あとは a タグ内で関数 ifr を呼び出して、引数にパズルの部屋の URL を指定してやっています。 クリックすると、パズルの部屋に飛びます。