

js-cafe5: ブラウザを自在に操る ～ブラウザオブジェクト

荻 幸旗

テキトーに自己紹介



名前：荻 幸旗

twitter: オギ コーキ @kor2_ogi

- 某IT系の営業マン 特技はテレアポです。瞬殺です。
- 中国に8年滞在、得意言語は中国語です。プログラム言語ですらありません。
- 考古学者目指してたので甲骨文読みます。（ソースコード読めた方がよかったです）



- プログラミングは去年の10月くらいからはじめたド素人です。
- あんまりいじめないでください。
- 補足等ガンガンしてくれると助かります！

僕、今回こんな事担当しました

- 3-1：イベントが発生したら処理を行なうイベントドリブンモデル
- 3-2：ブラウザオブジェクトの基本を押さえる



3-1

- クライアントサイドJSとイベントドリブンモデルを理解する
- イベントとイベントハンドラを関連づける
- イベントの標準の挙動をキャンセルする

クライアントサイドJSとイベントドリブンモデルを理解する

- 今まで勉強してきたようにjsはいろんな環境で動きます。
- 今回勉強するのはクライアントサイドjs！つまり、Webブラウザで使用するJavaScriptです
- 当たり前だけどブラウザ環境でないと動かないからご注意を！

イベントドリブンモデル

- ブラウザーページ上ではボタンをクリック、テキストボックスの内容を変更した、等様々なイベントがあります。
- そんなイベントに応じて実行するコードを記述する、プログラミングモデル！
- それがイベントドリブンモデルなんです！

イベントハンドラ

- イベントに対して処理内容を定義するコードの塊（関数）の事。

イベント

onBlur

ページやフォーム要素からフォーカスが外れた時に発生

onFocus

ページやフォーム要素にフォーカスが当たった時に発生

onChange

フォーム要素の選択、入力内容が変更された時に発生

onSelect

テキストが選択された時に発生

onSelectStart

ページ内の要素が選択されようとした時に発生 ※IEのみ

onSubmit

フォームを送信しようとした時に発生

onReset

フォームがリセットされた時に発生

イベント

onAbort

画像の読み込みを中断した時に発生

onError

画像の読み込み中にエラーが発生した時に発生

onLoad

ページや画像の読み込みが完了した時に発生

onUnload

ウィンドウを閉じた時、他のページに切り替えた時、
ページをリロード（更新）した時に発生

onClick

要素やリンクをクリックした時に発生

onDbClick

要素をダブルクリックした時に発生

onKeyUp

押していたキーをあげた時に発生

イベント

onKeyDown

キーを押した時に発生

onKeyPress

キーを押してる時に発生

onMouseOut

マウスが離れたした時に発生

onMouseOver

マウス乗った時に発生

onMouseUp

クリックしたマウスを上げた時に発生

onMouseDown

マウスでクリックした時に発生

onMouseMove

マウスを動かしている時に発生

onDragDrop

マウスでドラッグ&ドロップした時に発生 ※NN4のみ



イベントとイベントハンドラを関連づける

★イベントドリブンモデルでは以下の3つを定義する

- どの要素で？
- どのイベントを？
- どのイベントハンドラに関連付けるか？

①タグ内の属性として作成する

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<meta http-equiv="Content-type" content="text/javascript" />
<title>サンプル</title>
</head>
<body>

<script type ="text/javascript">
<!--
function btn_onclick(){
    window.alert('荻がクリックするとアラートができるコードを書くのに成功しました');
}
//-->
</script>

<input type="button" value="ダイアログ表示" onclick="btn_onclick()" />
</body>
</html>
```

ボタンを押したタイミングで警告ダイアログを表示する

- タグ内でイベントハンドラを宣言するには以下のように記述します。

タグ名 `on`イベント名="JavaScriptのコード"

- ごく簡単な処理の場合は直接記述も可、多用はおすすめしません。

```
<input type="button" value="ダイアログ表示"
onclick="window.alert('荻がクリックするとアラートができるコードを書く
のに成功しました');"/>
```

タグ内でスクリプトを記述する場合はデフォルトのスクリプト言語を<head>タグ配下の<meta>タグで宣言しましょう！

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-script-type"
content="text/javascript" />
</head>
```

②jsのコード内で宣言する ～プロパティとして設定～

- 本来レイアウトの定義をするHTMLの中にJSのコードを記述するのは良くない！

関連づけとイベントハンドラそのものの宣言を
まとめてjsコード内に記述できます！

```
<script type ="text/javascript">
<!--
//ページロード時に実行されるイベントハンドラを登録

window.onload = function (){
//ボタン(btn)クリック時に実行されるイベントハンドラを登録

document.getElementById('btn').onclick = function(){
    window.alert('荻がクリックするとアラートができるコードを書くのに成功しました');

};

};

//-->
</script>

<input id="btn" type="button" value="ダイアログ表示"/>
```

構文：イベントハンドラの登録①匿名関数

```
window.<onイベント名> = function (){..処理..}
```

↑<body>タグに対する設定

```
document.getElementById('ID値').<onイベント名> = function() {..処理..}
```

↑その他要素に対する設定

構文：イベントハンドラの登録②関数

```
window.<onイベント名> = 関数名 ←<body>タグに対する設定
```

```
document.getElementById('ID値').<onイベント名> = 関数名
```

↑その他要素に対する設定

★★イベントハンドラは性質上複数箇所使い回す事がほとんどないので、①のやり方がおススメです。★★

記述する場合の注意点

- イベント名はすべて小文字で記述！
(window.onLoad) みたいなのはダメ！
- プロパティとして設定しているのは関数オブジェクト！

関数オブジェクトであって、呼び出しではない！

window.onLoad = init() →ダメ

window.onLoad = init →OK

- 個別要素のイベハソはonloadイベハソの配下に！

document.getElementById.～は基本onloadの下に記

述しよう！じゃないとエラーになっちゃうぞ！

getElementByIdは“指定された要素を取得するためのメソッド”だと覚えておこう



イベント標準の挙動をキャンセルする

- イベントハンドラの挙動：クリックしたらダイアログがでるとかこういった動作の事
- キャンセルのためににはイベハンの戻り値にfalseを返せばいいだけ★

```
<body oncontextmenu="return false">  
</body>  
</html>
```

3-2

- ・ ブラウザーオブジェクトとは
- ・ ブラウザーオブジェクトの階層構造
- ・ ブラウザーオブジェクトにアクセスするには

ブラウザオブジェクトとは

- ブラウザオブジェクトとは、ブラウザ操作のための機能を集めたオブジェクト郡
- 古くからInternet ExplorerやNetscape Navigatorをはじめとした多くのブラウザで実装されているものです。
- 特にこれといった標準の規格が存在するわけではない。ブラウザベンダー個々で独自に実装されたローカルなオブジェクト郡です。
- そのため、過去はクロスブラウザ問題が取りざたされることも多かったのですが、昨今ではブラウザ同士の歩み寄りによってかなり解消されつつあります。もちろん、すべての機能が完全に互換性を持っているわけではありませんが、現在では主要なオブジェクトはそれなりに互換性を持っています。

ブラウザーオブジェクトの階層構造

- クライアントサイドJavascriptオブジェクトの階層構造で最上位に位置するのが、Windowオブジェクトです。
- クライアントサイドのオブジェクトは全て、このWindowオブジェクトを通じてアクセスします。
- その個々のクライアントサイドオブジェクトは、他のオブジェクトのプロパティとしてアクセスできます。

documentオブジェクトは、最も頻繁に使用するオブジェクトで、画面上に表示されているページに関するあらゆる情報(文字列、画像、フォーム、リンクなど)を持っています。

グローバルオブジェクト window オブジェクト

- ・最上位に位置する。
- ・クライアントサイドJavaScriptが起動するタイミングで自動的に生成される。
- ・グローバル変数やグローバル関数にアクセスするための手段を提供します。
- ・まさにクライアントサイドJSのグローバルオブジェクト

parent、**self**、**top**、
任意のウィンドウ名
(Windowオブジェクト)

screen
(Screenオブジェクト)

document
(Documentオブジェクト)

location
(Locationオブジェクト)

navigator
(Navigatorオブジェクト)

history
(Historyオブジェクト)

forms
(フォームの集合=Formオブジェクト配列)

anchors
(アンカーの集合=Anchorオブジェクト配列)

images
(画像の集合=Imageオブジェクト配列)

elements
要素の集合)

URLを表すオブジェクト

履歴を表すオブジェクト

Button	CheckBox	FileUpload	Password	Radio	Reset	Select	Submit	Text	Textarea
--------	----------	------------	----------	-------	-------	--------	--------	------	----------

options
(オプションの集合)

DOM : Document Object Model

- 先ほどの図にあるように多くのオブジェクトがDocumentオブジェクトの下位階層に位置しています。
- クライアントサイドオブジェクトの階層構造のうち、このDocumentオブジェクトから下位の階層のことをドキュメントオブジェクトモデル（Document Object Model : DOM）といいます。
- DOMとはWebブラウザのクライアントサイドオブジェクトの階層構造のうち、Documentオブジェクト以下のことと指す
- DOMは、Javascriptを語る上で特に大事なオブジェクトになっていくので、なんとなく覚えておいてください。

ブラウザーオブジェクトにアクセスするには

- グローバルオブジェクトと同じく、クライアントサイドJSのwindowオブジェクトは基本開発者は意識する必要がありません！

```
document.write('こんにちわ');
```

上記例の用に、windowオブジェクトを省略して
直接documentオブジェクトからかきだしてOK!!



```
window.document.write('こんにちわ');
```

Windowオブジェクト経由で呼び出してはいけない



```
window.document.write('こんにちわ');
```

Windowオブジェクトの中のwindowプロパティは自分自身を参照します、
そこを経由するのはOK
だけどコードが長くなるだけで意味がないからやらなくてOK!

- `document`は`Document`オブジェクトを参照する
プロパティ
- 今までのように記述するときはすべてオブジェク
トではなく参照するためのプロパティで記述して
いるので混乱しないようにしてください！

✿ 以上！ 僕の発表はここまで.....！ 続きよろしく！