## はじめに

数年前であれば Flash を使用して表現していたインタラクティブな演出やアニメーションが、 ここ最近は Flash に対応していないスマートフォンやタブレットなどの多様化に伴って、その ほとんどが JavaScript で表現されるようになりました。

このような大きな潮流の中で注目を集めているのが「jQuery」です。複雑なコードになりがちな JavaScript を手軽に扱うことのできる jQuery は、今では Web クリエイターにとって必要不可欠な存在になっています。それにより、関連書籍も多数存在していますが、反面そのほとんどは開発者が一人で執筆しているため、プログラムを深く理解していることが前提の解説となっているものが多く、日常的にプログラミングを行っていないデザイナーにとってはとっつきにくい内容になっているものが多いのが現状です。

そこで本書では「プログラム経験の少ないデザイナーでも容易に理解できる構成」を第一の目標として執筆を行いました。技術的な側面の解説やプログラミング手法、jQuery の詳細解説については、現場の第一線で日々業務を行っている経験豊富なプロフェッショナルの開発者が執筆を担当していますが、その一方で、執筆期間中はjQuery の知識が全くないデザイナーや、プログラミング未経験者を執筆チームに加えることで、これからjQuery を学ぶ人が「どこにつまづき」、「何に疑問を抱くのか」を常に検証し、そしてブラッシュアップしていきました。こういった日々の作業によって、確かな技術書として、より多くの人にとって最良の書籍となることを目指しました。この点は本書の強みの1つです。

また、Web デザイン関連の書籍においては、学習過程で使用するサンプルのクオリティも非常に大切だと考えています。そのため本書では、ただ単に解説を補足するような単純すぎるサンプルの掲載を避け、多くの人が「作ってみたい」、「仕事でも使えそうだ」と感じるようなサンプルであることを目指しました。本書内で使用するサンプルはすべて、第一線で活躍するアートディレクター監修の元、現場で多くのプロモーションサイトを制作しているインタラクションデザイナーが制作しています。ぜひ、実際に手を動かしてこれらのサンプルを制作し、jQueryをより深く習得していただけると嬉しいです。

今後、世の中のデジタル化が進むことは間違いありません。そして、その社会を作っていくうえで重要な役割を担うのは、みなさんをはじめとするクリエイター達です。本書が、みなさんがこれから制作していくであろうサイトや作品の一助となり、新しい表現がWeb上に登場することを心より祈っています。

最後に、通常の案件業務がありながらも、社内をマネジメントしてくれた仲村くん、素晴らしいサンプルを作ってくれた許くん、奥村さん、そして、執筆を担当した、安友さん、丈さん、鈴木慶太郎くん、本当にお疲れさまでした。そして執筆チームの面倒を丁寧にみてくださった SB クリエイティブの岡本晋吾さん、そして、何よりこの本を手に取ってくれた読者の方に感謝いたします。

2013 年 11 月 株式会社シフトブレイン 代表取締役 加藤琢磨



## Contents

本書のサンプルデータ	

Chapt		jQuery の基礎知識と本書の特徴
	01	iQuery の基礎知識と本書の特徴
\ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \		

13

01-01	jQuery の基礎知識	1
	• jQuery とは	1
	• jQuery の特徴と広く利用されている理由	…1
	Column ライブラリ、プラグイン、関数、拡張ライブラリの違い・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・	2
01-02	本書の目的と構成・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・	2
	• 本書の目的と特徴的な解説方法	2
	• 本書の構成	2
01-03	jQuery の実行環境の準備····································	2
	• サンプルファイルのダウンロード	2
	• jQuery のダウンロード	3
01-04	デバッグ方法	3
	• Chrome を使用したデバッグ	3
	- C-f: ナ	_

Chapter	02	
	02	

# 3ステップではじめるかんたん jQuery入門

35

02-01	体験することからはじめよう!30
	• サンプルファイルについて ······36
	• 本書の解説の流れ ······38
	Column   JavaScript の記述場所・・・・・・38
02-02	STEP 1 CSS のスタイルを変更してみよう

	• コードの解説 ····································
	• サンプルコードを書き替えてみよう!40
	Column         数値と文字列の違いに注意する・・・・・・・・・・・42
	• jQuery の最も基本的な構文の解説 ······42
	• HTML の準備が整うのを待つ仕組み·······46
	• ここまでのまとめ ·······48
	Column         CSS() メソッドに関する追加説明 ·······49
02-03	STEP 2 タイミングをコントロールしてみよう51
	• コードの解説 ······52
	• サンプルコードを書き替えてみよう!54
	•メソッドチェーンとは57
	• ここまでのまとめ ······58
02-04	STEP 3 アニメーションさせてみよう
	• コードの解説 ······59
	• サンプルコードを書き替えてみよう!61
	• その他の主な引数 ······62
	Column さまざまなイージング・・・・・・・・・・・64
	• animate() メソッドの仕様に関する注意点 ······66
	• ここまでのまとめ ······68

# 03

# jQuery の基本的な書き方

69

ナーダで恰削りつさまさまな八礼物
• オブジェクトとは ······70
• 変数とは ······71
●配列とは······75
●特殊な入れ物「this」······77
特定の機能をまとめる「関数」82
●関数とは·····82
•用意されている関数とオリジナルの関数83
• 関数の作り方 ······83
• 関数の呼び出し方84
●無名関数85
Column         無名関数を変数に格納する・・・・・・86

,

03-03	スコーフと命名規則・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・
	•スコープ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・
	•変数名と関数名の命名規則8
	Column 変数や関数 (メソッド) の返り値を確認する方法8
03-04	<b>演算子</b>
	• 算術演算子
	•代入演算子
	•比較演算子····································
	• 論理演算子
03-05	主な制御文・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・
	• 条件分岐 if 文 ··································
	• 条件分岐 switch 文 ···································
	•繰り返し処理 for 文 ···································

NaQ*	al	
Char		
	()	4 🅽

## 確かな基礎力を養う jQuery の基本テクニック

99

04-01	さまざまなホバーエフェクトで学ぶ
	動きのある表現の基本
	• ホバーエフェクトの HTML · · · · · · 10
	•ホバーエフェクトの CSS 10
	• 1 行目のボタンの JavaScript (jQuery) ······ 10
	•2 行目のボタンの JavaScript (jQuery) ······ 10
	•3 行目のボタンの JavaScript (jQuery) ······ 10
04-02	画像とキャプションの表現
	● キャプション表現の HTML · · · · · · 10
	• キャプション表現の CSS (共通部分) ······ 11
	• 1 つめの画像のキャプション表現
	Column filter() メソッドと find() メソッドの違い · · · · · · 11
	• 2 つめの画像のキャプション表現
	•3 つめの画像のキャプション表現
04-03	丸いボタンのレイアウト11
	• 丸いボタンの HTML ······· 11
	• 丸いボタンの CSS ······ 11
	•丸いボタンの JavaScript (jQuery) ······· 12

04-04	見え隠れするサイトハー
	• サイドバーの HTML 123
	• サイドバーの CSS · · · · · 124
	•サイドバーの JavaScript (jQuery) ······ 125
	● button 要素がクリックされた際に最初に実行される処理 ··········· 126
	• if 文による分岐処理 ······ 126
	Column toggleClass()、addClass()、removeClass() メソッド・・・・・・・128
04-05	タイポグラフィの表現 129
	● タイポグラフィ表現の HTML ······ 129
	• タイポグラフィ表現の CSS ······ 130
	• タイポグラフィ表現の JavaScript (jQuery) ······ 130
	<ul><li>プラグインの使い方</li></ul>
	Column プラグイン 『jquery.typoshadow.js』 の処理内容 · · · · · · · · · · · 132

# jaaptel 05

## 活用の幅を広げる jQuery の必修テクニック

05-01	Basic 滑らかな動きのスライドショー 135
	• スライドショーの概要······135
	<ul><li>■スライドショーの処理の流れ・・・・・・・・・・135</li></ul>
	• スライドショーの HTML と CSS 136
	•スライドショーの JavaScript (jQuery) ······ 137
	●変数の準備
	●スライドショーの実行······140
	Column JavaScript が無効になっている場合を考慮する
05-02	Advanced 多機能なスライドショー
	<ul><li>スライドショーの概要·······144</li></ul>
	<ul><li>スライドショーの概要・・・・・・・・144</li><li>スライドショーの HTML・・・・・145</li></ul>
	• スライドショーの HTML
	<ul><li>スライドショーの HTML 145</li><li>スライドショーの構造 146</li></ul>
	<ul> <li>スライドショーの HTML 145</li> <li>スライドショーの構造 146</li> <li>スライドショーの CSS 147</li> </ul>
	• スライドショーの HTML145• スライドショーの構造146• スライドショーの CSS147• スライドショーの JavaScript (jQuery)150
	• スライドショーの HTML145• スライドショーの構造146• スライドショーの CSS147• スライドショーの JavaScript (jQuery)150• 変数の準備153

	<ul><li>任意のスライドを表示する関数 goToSlide() 関数</li><li>ナビゲーションとインジケーターの</li></ul>	156
	状態を更新する関数 updateNav() 関数 ······	158
	<ul><li>クリックイベントに処理を登録する</li></ul>	
	タイマーの開始と一時停止 startTimer() 関数と	
	stopTimer() 関数 ···································	162
	<ul><li>スライドショーの開始····································</li></ul>	
	Column タイマーの仕組み · · · · · · · · · · · · · · · · · · ·	
05-03	Basic スティッキーヘッダー	165
	• スティッキーヘッダーの概要	165
	<ul><li>処理の流れ ····································</li></ul>	
	•スティッキーヘッダーの HTML と CSS ······	166
	•スティッキーヘッダーの JavaScript (jQuery) ······	
	•変数の準備	
	• ウィンドウのスクロールイベントを監視する処理	
	• ウィンドウのスクロールイベントを発生させる処理	
05-04	Advanced デザインが変化するスティッキーヘッダー	173
	<ul><li>スティッキーヘッダーの概要····································</li></ul>	
	<ul><li>スティッキーヘッダーの仕組み</li></ul>	174
	•スティッキーヘッダーの HTML と CSS ······	
	•スティッキーヘッダーの JavaScript (jQuery) ······	
	• 変数の準備	
	• HTML 要素の挿入····································	
	• ウィンドウがスクロールされた際の処理	
05-05	Basic 画面領域を有効活用できるタブ	
	<ul><li>タブの仕組み</li></ul>	
	• タブの HTML····································	
	• タブの CSS	
	• タブの JavaScript (jQuery)·······	
	<ul><li>変数の準備</li></ul>	
	<ul><li>タブがクリックされたときの処理 ····································</li></ul>	
05-06	Advanced 高機能で拡張しやすいタブ	
	<ul><li>タブの仕組み</li></ul>	
	• jQuery UI とは・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・	
	• タブの HTML······	
	• タブの CSS	191
	• タブの JavaScript (jQuery)······	192
	亦料の准件	100

	aiOuary III Taba OH+tht/teu+
	• jQuery UI Tabs の基本的な使い方
	• ハイライトの位置を調整する関数 moveHighlight() 関数 · · · · · · 194
05-07	Basic スムーズスクロール 196
	• スムーズスクロールの仕組み ····· 196
	• スムーズスクロールの HTML と CSS · · · · · 197
	•スムーズスクロールの JavaScript (jQuery) ······ 197
	<ul><li>◆クリックイベントへの処理の登録 ······ 198</li></ul>
	• scrollTop プロパティを利用できる要素の検出 · · · · · · · · 199
05-08	Advanced 拡張性のあるスムーズスクロール 200
	• スムーズスクロールの仕組み ····· 200
	• スムーズスクロールの HTML と CSS · · · · · · 201
	•スムーズスクロールの JavaScript (jQuery) ······ 202
	<b>Column</b> 任意のページ内リンクでスムーズスクロールを有効にする方法 202
1	
(06)	高機能なギャラリーページを作ってみよう! 203

06-01	ギャラリーページの全体像
	<ul><li>ギャラリーページの主な機能 204</li></ul>
	•本章の解説の流れ
06-02	Basic シンプルなギャラリーページの作成 206
	•動作確認に関する注意点
	•2つの JavaScript ライブラリ
	• ギャラリーページの HTML と CSS 208
	<ul><li>ギャラリーページの元データの構造</li></ul>
	• ギャラリーページの JavaScript (jQuery) ······ 211
	• このサンプルの処理対象となる HTML 要素 ······ 212
	• Masonry の準備······212
	• \$.getJSON() メソッドによる JSON データの取得 ······ 213
	• \$.each() メソッドによるループ処理 ······ 213
	• HTML 文字列の生成·······215
	• Masonry による各要素のレイアウト······ 217
	Column         JavaScript による文字列の記述方法・・・・・・・・・・219
06-03	Advanced フィルタリング機能を持つ
	ギャラリーページの作成 220

.

		<ul><li>関数の分離 ····································</li></ul>	22
		• Basic 版との主な違い	22
		• ギャラリーサイトの HTML	
		• ギャラリーサイトの CSS	22
		• ギャラリーサイトの JSON	22
		●ギャラリーを初期化する関数 initGallery() 関数 ······	22
		• アイテム群を生成し、ドキュメントに挿入する関数	
		addItems() 関数 ······	22
		Column 正規表現の基本・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・	23
		• フィルタリング機能の仕組み	23
		●アイテムのフィルタリング機能 filterItems() 関数······	23
		• ラジオボタンのカスタマイズ	23
	06-04	Advanced 選択画像の拡大機能とキャプションの追加・	23
		<ul><li>◆今回追加する機能の概要····································</li></ul>	
		• Colorbox のダウンロードと読み込み ·······	
		• Colorbox の実行······	
	06-05	Advanced マウスの移動方向による	
	00-05	ホバーエフェクト機能の追加	24
		<ul><li>今回追加する機能の概要····································</li></ul>	
		● ホバーエフェクトの HTML · · · · · · · · · · · · · · · · · · ·	
		• ホバーエフェクトの CSS ··································	
		ホバーエフェクトの JavaScript (jQuery)	
		オーバーレイのアニメーション_ hoverDirection() 関数	
		Column         通常の関数と即時関数の違い	
		マウスの方向を検出する getMouseDirection() 関数	
		* イプスの万円を採出する SetiviouseDirection() 因数	24
apter			
Jac	07	·	
	07	jQuery を活用したさまざまな表現	249
	07-01	画像の読み込みのプログレス表示機能	
		<ul><li>プログレス表示機能の概要 ····································</li></ul>	
		• プログレス表示に必要な処理	
		• プログレス表示の HTML と CSS······	
		• プログレス表示の JavaScript (jQuery) ······	
		•表示に必要な要素を jQuery オブジェクト化 ····································	25

	● 画像の読み込み状況の監視とフロクレス表示への反映 255
	• 画像の読み込み状況をプログレス表示に反映する updateProgress()
	関数256
	● プログレス表示の終了処理 ······258
07-02	画像シーケンスのアニメーション 260
	●画像シーケンスのアニメーションの概要 260
	●アニメーションの仕組み······261
	• 画像シーケンスのアニメーションの HTML 261
	• 画像シーケンスのアニメーションの CSS 262
	•画像シーケンスのアニメーションの JavaScript (jQuery) ······ 262
	●使用する変数の準備
	•マウスホイールイベントの取得
	●アニメーションの開始 startAnimation() 関数 ······ 266
	• アニメーションの内容 animateSequence() 関数 ······ 267
	●アニメーションの終了 stopAnimation() 関数······ 269
	• 全画面表示
07-03	回転のアニメーションによる
	インフォグラフィック 272
	•インフォグラフィックとは······272
	<ul><li>今回のサンプルのポイント 273</li></ul>
	•パイチャートの仕組み······ 273
	• インフォグラフィックの HTML
	• インフォグラフィックの CSS ······ 275
	• transform プロパティのアニメーション · · · · · · · 277
	•インフォグラフィックの JavaScript (jQuery) ······ 278
	●各チャートの処理-準備
	●各チャートの処理-角度の操作
07-04	マスクのアニメーション284
	<ul><li>◆今回のサンプルの仕組み·······284</li></ul>
	• マスクのアニメーションの HTML 285
	•マスクのアニメーションの CSS · · · · 286
	• animate() メソッドで CSS の clip プロパティを操作する方法 ······ 287
	•マスクのアニメーションの JavaScript (jQuery)······ 287
	•マスクのアニメーションの JavaScript (jQuery)       287         •マスクのアニメーション       290



# jQuery リファレンス

• Core ーコア・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・	292
• Selectors ーセレクタ ······	293
• Traversing -検索 ·······	296
• Attributes/CSS -属性と CSS ··································	298
● Manipulation 一操作·······	300
●Events ーイベント・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・	303
• Effects ーエフェクト ·······	304
• Ajax ·····	305
• Utilities ーユーティリティ ·······	306
• 本書で使用している外部ファイル	307
Normalize.css	307
Modernizr	308
• HTML5 Shiv·····	309
<del></del>	

37

## 体験することからはじめよう!

前章でも触れましたが、jQuery はとてもシンプルな JavaScript ライブラリであるため、初心者の方でも要点さえ理解すればすぐに使用できるようになります。しかしその反面、多機能でもあるため、基礎知識がない状態で膨大なリファレンスを先頭から読み進めたり、Web 上に散らばっている断片的な情報を拾い読みする方法だけでは、「何をどう使えば良いのか」、「何から覚えたら良いのか」がわかりにくいという一面もあります。初心者の方が効率良くjQuery を習得するためには、適切な順番でjQuery の機能や使用方法を学んでいくことが必要です。そこで、まずは以下の3つのステップで実際に手を動かしながら、jQuery の基礎とそれに関連するJavaScript の基礎を学んでいきましょう。

STEP 1 CSS のスタイルを変更してみよう

STEP 2 タイミングをコントロールしてみよう

STEP 3 アニメーションさせてみよう

この3つのステップではいずれもシンプルな機能しか利用しませんが、これらは jQuery における「見た目の表現」に関する基礎となります。そのため、読み進めていくと表現の幅が広がっていく感覚を実感できると思います。

## サンプルファイルについて

本章では、前節でダウンロードした本書のサンプルファイルのうち、[Chapter02] フォルダ 内のファイルを使用します。フォルダ内の index.html をブラウザで開くと以下のようなシン プルなページが表示されます。

## Fig index.html

36

Creative jQuery Sample

# **Creative jQuery**

COPYRIGHT © SHIFTBRAIN IN

index.html をテキストエディタなどで開き、内容を確認しておきましょう。

```
html サンプルファイルの HTML
                                                                      index.html
 <!DOCTYPE html>
 <html lang="ja">
 <head>
 <meta charset="UTF-8">
 <meta http-equiv="X-UA-Compatible" content="IE=edge">
 <title>Creative jQuery</title>
 <link rel="stylesheet" href="./css/normalize.css">
 <link rel="stylesheet" href="./css/main.css">
 <script src="./js/vendor/jquery-1.10.2.min.js"></script>
 <script src="./js/vendor/jquery-ui-1.10.3.custom.min.js"></script>
 <script src="./is/main.is"></script>
 </head>
 <body>
 <header class="page-header" role="banner">
    <h1>Creative iQuery Sample</h1>
 </header>
 <div class="page-main" role="main">
    <div id="tvpo">
         <div class="inner">Creative jQuery</div>
    </div>
 </div>
 <footer class="page-footer" role="contentinfo">
    <small class="copyright">COPYRIGHT &copy;
         <a href="http://www.shiftbrain.co.jp" target=" blank">
             SHIFTBRAIN Inc.</a>
    </small>
 </footer>
 </body>
 </html>
```

head 要素内で複数の CSS ファイルを読み込んでいますが ①、normalize.css はクロスブラウザ対応のためのファイルです。このファイルについては P.307 を参照してください。本章を読み進めるうえでは特に詳細を把握しておく必要はありません。

また、3つの JavaScript ファイルを読み込んでいますが②、これらのうち、本章でみなさんが使用するのは main.js のみです\*¹。main.js は、みなさんが本書を読み進めながらプログラムを記述していくために用意した空のファイルです (何も記述されていません)。

<sup>\*1</sup> main.jsよりも先に (上部で) 読み込んでいる2つのファイルは、jQuery のファイルと jQuery の拡張ライブラリである [jQuery UI]です。 jQuery UI については、これを使用するタイミングで別途説明します。

本章では、基本的に以下の流れで解説を進めていきます。

- (1) main.is をエディタで編集する
- (2)編集後にブラウザで開き、動作確認を行う

書籍を読むだけ、サンプルを見るだけではなかなか身につきません。そのため、本章のmain.jsは、あえて空っぽの状態でスタートします。入力するのが面倒だと感じる方もいるかもしれませんがそれがプログラムというものです。きちんと技術を身につけるためにも、実際に手を動かしてプログラムを書いていくことをお勧めします。それでははじめましょう!

## Point!

38

#### 本章のサンプルファイルの対応ブラウザ

本章のサンプルについては、IE8 などの古いブラウザではサポートされていない CSS プロパティが一部に含まれています。そのため、動作を確認する際はなるべく新しいブラウザを使用してください。なお、5 章以降で解説するサンプルでは一部の古いブラウザにも対応しています。



# JavaScript の記述場所

JavaScript は、HTML内に直接記述する方法と外部ファイルに記述して読み込む方法がありますが、本書では外部ファイル (main.js) に記述して読み込む方法を採用しています。

HTML に記述する方法は手軽ではあるのですが、HTML と JavaScript が 1 つのファイルにまとまってしまうため、Web サイトが複雑になればなるほど、管理性や更新性が損なわれてしまいます。現在では外部ファイルに記述して読み込む方法が一般的に採用されています。なお、外部ファイルに記述して読み込む方法の場合は、js ファイルの読み込む順番に注意が必要です。上記では、jQuery の js ファイルを先に読み込んでから、main.js を読み込んでいるのがわかると思います。このように、先に jQuery を読み込んでおかないと、main.js

のプログラムから jQuery を使用することはできません (エラーになります)。

STEP 1 CSS のスタイルを変更してみよう

まずは、jQueryを使用して CSS のスタイルを変更してみます。

main.jsをエディタで開き、以下のコードを追加してください。この段階ではコードの意味を理解する必要はありません。後で詳しく解説します。

# JavaScript CSS のスタイルの変更 (#typo の要素の変更) \$(function(){ \$('#typo').css('color', '#ebc000'); });

上記のコードを追加したら main.js を保存して、ブラウザをリロードします。すると下図のように #typo の要素の文字色が変わります $^{*1}$ 。

Chapter 02

39



## コードの解説

それでは先ほど追加したコードの内容を見ていきましょう。理解しやすいようにコードを以下の2つのパートに分けます。

<sup>\*1 #</sup>typo の要素が変化しない場合は、記述ミスをしている可能性があります。ブラウザの開発者ツールを起動してコンソールを確認してみましょう (P.32)。

main.js

\$('#typo').css('color', '#ebc000');

1つめのパートのコードの意味は後で説明します ( $\Rightarrow$  P.46) \*  $^{2}$ 。ここでは 2 つめのパートを見てください。CSS をご存じの方であれば、コードの細かい意味はわからなくても、どのような処理をしているのか想像できたのではないでしょうか。このコードの内容は以下の通りです。

#### Fig スタイルを変更するコードの内容

\$('セレクタ').css('プロパティ', '値');

### Table』書き換え可能な箇所

要素	説明
セレクタ	ここには CSS と同じセレクタを指定できる。サポートされていないものも多少あるが、 特定の要素を指定するのに困ることはない
プロパティ	CSS のプロパティを指定する
値	プロパティに設定する値を指定する

つまり、このコードは「**\*セレクタ"に該当する要素の CSS の "プロパティ"を "値"に変更する**」という処理を行います。そうです、CSS の知識をそのまま使用できるのです\*<sup>3</sup>。世の中のインタラクティブな動きのある Web サイトは、基本的にはこのように「JavaScript で CSS を随時書き替える」ことで実現されています\*<sup>4</sup>。このコードはそのほんの入口ですが、とても重要です。

## サンプルコードを書き替えてみよう!

本章のサンプルはHTMLの要素が少ないため、あまりいろいろな実験はできませんが、ここで実際に以下のような感じでセレクタやプロパティ、値を自由に書き替えてみてください。 HTML内の各要素をjQueryで操作できることがわかります。

## JavaScript。変更例①

main.js

```
$(function(){
    $('#typo .inner').css('transform', 'rotate(10deg)');
});
```

- \* 2 1 つめのパートのコードにはコメントアウトが含まれています。// 以降の文字列は行末までコメントアウトされます。複数行のコメントアウトは CSS と同じ /\* 《コメント》\*/ です。
- \*3 jQuery では、該当する各要素の style 属性の情報を書き換えることで CSS を適用します。そのため、 CSS で「!important」を使用している項目は、その設定が優先されるので注意してください。
- \* 4 Flash や CSS3、canvas などによる表現は除きます。

上記のコードを保存してブラウザをリロードすると以下のようになります。

#### Fig 変更例①の表示





このように、CSS3 プロパティである transform プロパティの rotate() 関数も問題なく使用できます\*5。

もう1つ例を見てみましょう。

上記では無意味に長いセレクタで [変更例①] と同じ #typo .inner を選択しています。ご覧の通り、jQuery は古いブラウザでは使えない:nth-child() などの CSS3 セレクタにも対応しています。ここでは opacity (要素の不透明度) を 0.5 (50%) に指定しています\*6。上記のコードを保存してブラウザをリロードすると以下のようになります。

#### Fig 変更例②の表示



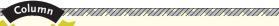


40

<sup>\*5</sup> transform プロパティには通常はベンダープレフィックスが必要ですが(執筆時点)、プロパティのベンダープレフィックス は jQuery が自動的に付加してくれます。ただし、「値」 にベンダープレフィックスが必要な場合は付加した文字列を渡す必要があります。

<sup>\* 6</sup> IE8 以下では、opacity(不透明度)の指定方法が異なりますが、jQuery が自動的に差異を吸収してくれるので統一的に「opacity」と  $0\sim 1$  の数値で指定できます。





## 数値と文字列の違いに注意する

上記でいくつかの指定例を示しましたが、値の指定方法が2通りあることに気づいたでしょ うか。最初の例では「css('color', '#ebc000')」のように値を「'」(シングルクォーテー ション) で囲っていますが、2 つめの例では [css('opacity', 0.5)] のように直接数値を 指定しています。これらの見た目はそれほど変わりませんが、プログラム的には大きく異なる ので注意してください。

iQueryでは「'」や「"」には以下のような役割があります。

重要「'」(シングルクォーテーション) または「"」(ダブルクォーテーション) で囲まれた箇 所は文字列として認識されます。そのため文字列以外、例えば数値を指定する場合は「'|や 「"|で囲まずに値を指定することが必要です。

JavaScript はとても寛容なプログラム言語なので、数値を指定すべき箇所に「'」を記述し て指定しても(文字列を指定しても)自動的に数値に変換して処理してくれる場合があるので、 必ずしもエラーになるわけではありませんが、エラーになる場合もあるので、なるべく上記の ルールを守り、数値を指定すべきところには数値を指定し、文字列を指定すべきところには文 字列を指定するように心がけてください。

## jQuery の最も基本的な構文の解説

CSS のスタイルを変更する方法が何となくわかった所で、コードの内容をもう少し掘り下げ て見ていきます。ここからは少しボリュームがありますが、少しずつ理解を深めていきましょう。 まずは先ほどパート分けした2つめのパートを使用して、iQueryの基本的な構文と考え方を解 説します。

JavaScript 2つめのパート

main.js

\$('#typo').css('color', '#ebc000');

これを iQuery 的に解釈すると「\$('#typo') というモノ(命令の対象)に対して、css ('color', '#ebc000')という命令を指示している(命令の内容)」と捉えることができます。 なぜこのような解釈になるのでしょうか。1つずつ見ていきましょう。

## 命令の内容:iQuerv メソッドと引数

まずは命令の内容の部分を見てみます。

#### JavaScript 命令の内容

css('color', '#ebc000');

コードの解説で触れた通り、css()では「()」(カッコ)の中に記述した 'color' が CSS の プロパティ、'#ebc000' がそのプロパティの値に相当し、「CSS の color プロパティに #ebc000 を指定せよ」という命令であることがわかります。

この css() という命令は、JavaScript に定義されている命令ではなく、iQuery 独自の命令 です。jQueryのjsファイルの中にcss()という命令が定義されており、その命令をmain.js から利用しています。このようなiQuery独自の命令のことを「jQueryメソッド」と呼びます(以 降は「メソッド」と記述します)。

また、css() の「()」の中に記述した CSS プロパティや値のように、メソッドに引き渡され るデータを「引数」と呼びます\*7。

まずは、上記のコードが jQuery のメソッド(css()) と 2つの引数('color', '#222') で構 成されているということを把握しておいてください。

#### Fig スタイルを変更するコードの内容

.css('color', '#222'); iQueryメソッド 2つの引数

## 命令の対象: \$() 関数と iQuery オブジェクト

さてここで css() というメソッドの「命令の対象」になっているのは「id="typo" が指定されて いる HTML 要素」ですが、iQuery では HTML 要素に対して直接 css() メソッドを実行するこ とはできません。メソッドを使用するためには、jQuery 独自の形式で命令の対象を指定するこ とが必要です。それが以下の部分になります。

#### JavaScript 命令の対象部分

main.js

\$('#typo')

<sup>\*7</sup> メソッドに引数を指定することを「引数を渡す」といい、メソッドを実行することを「メソッドを呼ぶ(呼び出す)」といいます。

**\$()** と CSS のセレクタ (上記では '#typo') の関係は、先述した css() メソッドと引数 (CSS プロパティおよび値)の関係と同じです。セレクタは \$() の引数として渡されています。

### Fig. 命令対象の内容

```
$('#typo');
関数 1つの引数
```

\$() は「HTML 要素をもとにして iQuery 独自の要素を生成する」という機能を持っている関数で す。HTML要素を指定する方法として、CSSのセレクタを指定できます。この\$() 関数によっ て iQuery の形式に変換された要素のことを「jQuery オブジェクト」と呼びます。

### メソッドと関数



index.html

\$() と css() は、引数を伴うという意味では類似していますが、\$() はメソッドとはいわ ず、「関数」といいます。この辺りが少しややこしいのですが、現時点では、命令文の前に「.」 (ドット)がついているものは「メソッド」、ついていないものは「関数」だと理解しておいてください。

html iQuery オブジェクトとして変換される要素(コード内の赤字の箇所)

SHIFTBRAIN Inc.</a>

</small>

今回使用したコードを用いて解説すると、「\$('#typo')」の指定によって index.html 内の 全 HTML のうち、以下の赤字の箇所の要素のみが iQuery オブジェクトに変換されます $\mathbf{0}$ 。こ のことから「\$('#typo')」に続いて指定されているメソッドは、この jQuery オブジェクトに 対して実行されると理解できます。

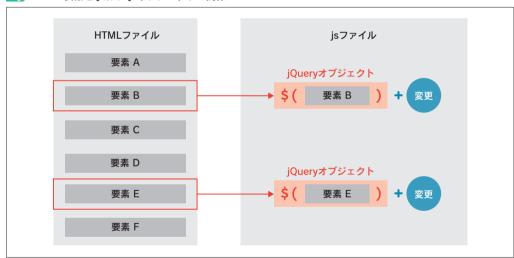
## //省略 <body> <header class="page-header" role="banner"> <h1>Creative jQuery Sample</h1> </header> <div class="page-main" role="main"> <div id="typo"> <div class="inner">Creative jQuery</div> </div> </div> <footer class="page-footer" role="contentinfo"> <small class="copyright">COPYRIGHT &copy; <a href="http://www.shiftbrain.co.jp" target="\_blank">

```
</footer>
</body>
//省略
```

他の例を挙げると、例えば「\$('h1')」を指定した場合は、以下の要素がiQuery オブジェク トに変換され、これが命令の対象になります2。

## html jQuery オブジェクトとして変換される要素 (コード内の赤字の箇所) index.html //省略 <body> <header class="page-header" role="banner"> </header> <div class="page-main" role="main"> //省略

#### Fig. HTML 要素と iQuery オブジェクトの関係



また、処理の流れも基本的に以下の通りです。

- (1) \$() 関数で命令の対象となる HTML 要素を jQuery オブジェクトに変換し、
- (2) その jQuery オブジェクトに対してメソッドを呼び出して変更を加える

jQueryには、css()メソッド以外にも、たくさんのメソッドが用意されていますが、どのメ ソッドの場合でもこの流れは変わりません。これが jQuery の基本です。ここで登場した各用語 の関係は次の図のように表されます。これらが iQuery において最も重要な登場人物になります ので、しっかりと関係性を把握しておいてください。

◯ Chapter ◯2 3 ステップではじめるかんたん jQuery 入門

# Fig 主な登場人物の関係図 jQueryオブジェクト

### 変更する命令

.

メソッド (「引数1」, 引数2」)

## 「.」(ドット)と「;」(セミコロン)の役割

**\$**( HTML要素

今回解説に使用しているコードでは、**\$()** 関数 (jQuery オブジェクト) に続けて css() メソッドが指定されていました。この関数とメソッドの間にある「・」(ドット) には jQuery オブジェクトと jQuery メソッドをつなぐ役割があります。

また、行末の「;」(セミコロン) は、JavaScript の「文」の終わりを表す記号です。日本語の文における「。」にあたるものです。いくつかの例外もあるのですが、現時点では JavaScript の 1 つの命令が終わったら「;」をつけると覚えておいてください。

#### Fig 「.」(ドット) と「;」(セミコロン) の役割



## HTML の準備が整うのを待つ仕組み

さて、後に回していたサンプルコードの1つめのパートについて解説します。

#### JavaScript 1つめのパート

main.js

\$(function(){
 //2つめのパートが記述されている
});

このコードの役割を確認するために、1つめのパートだけを削除した状態でブラウザをリロードしてみてください。するとスクリプトが動作していないことが確認できるはずです。なぜこうなってしまうのでしょうか。これにはHTMLの構造が関係しています。

もう一度サンプルの index.html のコードを見てみましょう。

#### html サンプルの HTML

index.html

```
<!DOCTYPE html>
<html lang="ja">
<head>
<meta charset="UTF-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<title>Creative jQuery</title>
<link rel="stylesheet" href="./css/normalize.css">
<link rel="stylesheet" href="./css/main.css">
<script src="./js/vendor/jquery-1.10.2.min.js"></script>
<script src="./js/vendor/jquery-ui-1.10.3.custom.min.js"></script>
<script src="./js/main.js"></script> •—
</head>
<body>
   //省略
</body>
</html>
```

ブラウザは基本的に、HTML を「コードが記述された順番通り」に処理していきます。つまりブラウザは、1 行目にある <!DOCTYPE html> から順に、html 要素、head 要素、<meta charset="UTF-8">…という順で認識していきます。

そして、このブラウザの処理が <script src="./js/main.js"></script> の部分に到達すると、ブラウザは main.js を読み込みます ①。そして、main.js の読み込みが完了するとその内容をすぐに実行します。

この際、main.jsにはその後に続く HTMLの要素を操作する命令が書かれています。しかし ブラウザはこの時点ではまだ head 要素の途中までしか認識していないため(それ以降の要素は まだ認識していないため)、処理を実行することができません。

この問題を解決するための仕組みが、1つめのパートにあります。以下のように \$(function (){ }); を記述すると、ブラウザはその中に記述された処理をいったん**予約状態**にしておき、HTML を最後まで読み込んだ時点でそれらを実行するようになります。

#### JavaScript ブラウザが HTML を最後まで読み込むのを待つための仕組み

```
$(function(){
    //HTMLがロードした後に実行する処理
});
```

jQuery を利用したプログラムでは、最初に HTML の要素を選択することがとても多いため、 ほぼすべてのコードが上記のような構成になります。本書でもすべてのサンプルにこのコードが 記述されています。

46

## ここまでのまとめ

ここでいったんこれまでの解説を振り返ってみます。これまでの解説で以下のことを学びまし た。これらの内容はすべて jQuery の最も基本的な項目であり、いずれもとても重要です。現時 点では概要的な内容しか解説していない箇所もありますが、しっかりと理解してから次に進んで ください。

- iQuery で CSS のスタイルを変更する方法
- 数値と文字列の違い
- jQuery メソッドや \$() 関数、jQuery オブジェクトの概要と関係性
- [.](ドット) と[;](セミコロン) の役割
- HTML の準備が整うのを待つ仕組み

これらについてきちんと理解できているか振り返ってみてください。なお、次項の解説も上記 と同様に以下の流れで学習していきます。

- (1) main.js にサンプルコードを記述する
- (2) ブラウザで index.html の表示内容を確認する
- (3)記述したサンプルコードの解説
- (4) コードを自由に書き替えてみる
- (5) ブラウザで index.html の表示内容を確認する
- (6) サンプルコードを使用した jQuery の基本解説