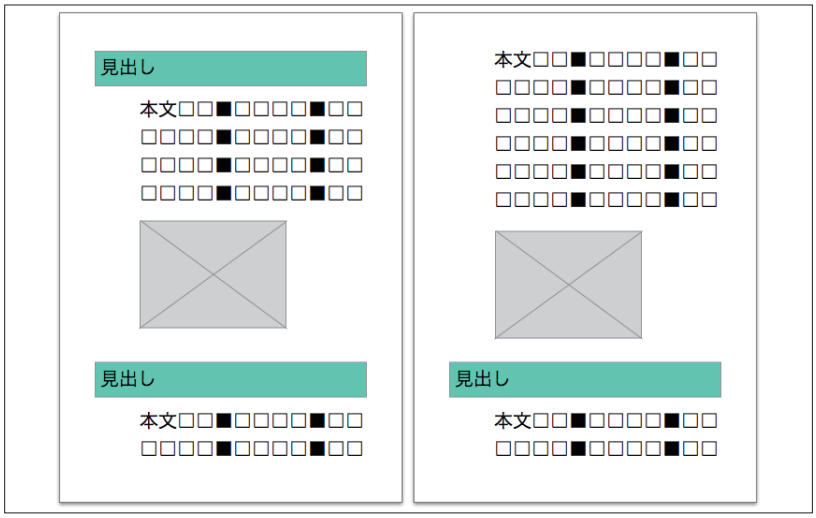
中級者向けの IT 書でもっとも多いデザインは、「一段組みで文章の途中に図版やソースコードを挟みつつ、章末またはセクション末までリニアに続いていくタイプ」です。このタイプがより速く組めるようになれば、その分を校正期間に回してクォリティアップを図れるようになります。



一段組みリニア型のイメージ

このタイプのデザイン指定は、たいていの場合、「各パーツのサイズ」と「パーツ間隔」の指定になりますが、それをそのままフォーマットにしてはいけません。
そのまま各パーツがバラバラの状態で組んでしまうと、後でテキストの追加や削除が発生したときに、それ以降を手作業でずらすことになってしまいます。

![デザイン指定そのままのフォーマットでは修正が大変](img/fig2.png)

では、組み直しが発生しない完璧な原稿をもらえばいい……というのは組む側の都合であって、作っているものが「解説書」である以上、デザインが複雑だから、DTP が大変だから、間違いがあっても直さないというのは本末転倒です。組みやすいフォーマットを作ることで、テキストや画像追加ぐらいなら楽勝！という状態に持って行きましょう。

中級者向けの IT 書でもっとも多いデザインは、「一段組みで文章の途中に図版やソースコードを挟みつつ、章末またはセクション末までリニアに続いていくタイプ」です。このタイプがより速く組めるようになれば、その分を校正期間に回してクォリティアップを図れるようになります。

図 1：一段組みリニア型のイメージ

このタイプのデザイン指定は、たいていの場合、「各パーツのサイズ」と「パーツ間隔」の指定になりますが、それをそのままフォーマットにしてはいけません。

そのまま各パーツがバラバラの状態で組んでしまうと、後でテキストの追加や削除が発生したときに、それ以降を手作業でずらすことになってしまいます。

![デザイン指定そのままのフォーマットでは修正が大変](img/fig2.png)

では、組み直しが発生しない完璧な原稿をもらえばいい……というのは組む側の都合

中級者向けの IT 書でもっとも多いデザインは、「一段組みで文章の途中に図版やソースコードを挟みつつ、章末またはセクション末までリニアに続いていくタイプ」です。このタイプがより速く組めるようになれば、その分を校正期間に回してクォリティアップを図れるようになります。

図 1 一段組みリニア型のイメージ

このタイプのデザイン指定は、たいていの場合、「各パーツのサイズ」と「パーツ間隔」の指定になりますが、それをそのままフォーマットにしてはいけません。

図 1 一段組みリニア型のイメージ

そのまま各パーツがバラバラの状態で組んでしまうと、後でテキストの追加や削除が発生したときに、それ以降を手作業でずらすことになってしまいます。

![デザイン指定そのままのフォーマットでは修正が大変](img/fig2.png)

では、組み直しが発生しない完璧な原稿をもらえばいい……というのは組む側の都合であって、作っているものが「解説書」である以上、デザインが複雑だから、DTP が大変だから、間違いがあっても直さないというのは本末転倒です。組みやすいフォーマットを作ることで、テキストや画像追加ぐらいなら楽勝！という状態に持って行き

中級者向けの IT 書でもっとも多いデザインは、「一段組みで文章の途中に図版やソースコードを挟みつつ、章末またはセクション末までリニアに続いていくタイプ」です。このタイプがより速く組めるようになれば、その分を校正期間に回してクォリティアップを図れるようになります。
このタイプのデザイン指定は、たいていの場合、「各パーツのサイズ」と「パーツ間隔」の指定になりますが、それをそのままフォーマットにしてはいけません。

図 1 一段組みリニア型のイメージ

そのまま各パーツがバラバラの状態で組んでしまうと、後でテキストの追加や削除が発生したときに、それ以降を手作業でずらすことになってしまいます。

![デザイン指定そのままのフォーマットでは修正が大変](img/fig2.png)

では、組み直しが発生しない完璧な原稿をもらえばいい……というのは組む側の都合であって、作っているものが「解説書」である以上、デザインが複雑だから、DTP が大変だから、間違いがあっても直さないというのは本末転倒です。組みやすいフォーマットを作ることで、テキストや画像追加ぐらいなら楽勝！という状態に持って行きましょう。

中級者向けの IT 書でもっとも多いデザインは、「一段組みで文章の途中に図版やソースコードを挟みつつ、章末またはセクション末までリニアに続いていくタイプ」です。このタイプがより速く組めるようになれば、その分を校正期間に回してクォリティアップを図れるようになります。

図 1 一段組みリニア型のイメージ

このタイプのデザイン指定は、たいていの場合、「各パーツのサイズ」と「パーツ間隔」の指定になりますが、それをそのままフォーマットにしてはいけません。

図 2 一段組みリニア型のイメージ

図 3 一段組みリニア型のイメージ

そのまま各パーツがバラバラの状態で組んでしまうと、後でテキストの追加や削除が発生したときに、それ以降を手作業でずらすことになってしまいます。

![デザイン指定そのままのフォーマットでは修正が大変](img/fig2.png)

では、組み直しが発生しない完璧な原稿をもらえばいい……というのは組む側の都合であって、作っているものが「解説書」である以上、デザインが複雑だから、DTP が大変だから、間違いがあっても直さないというのは本末転倒です。組みやすいフォー

このタイプのデザイン指定は、たいていの場合、「各パーツのサイズ」と「パーツ間隔」の指定になりますが、それをそのままフォーマットにしてはいけません。

図２ 一段組みリニア型のイメージ

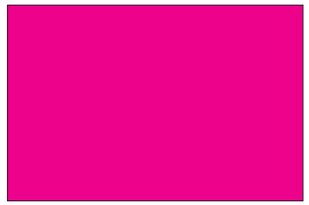


図３ 一段組みリニア型のイメージ

そのまま各パーツがバラバラの状態で組んでしまうと、後でテキストの追加や削除が発生したときに、それ以降を手作業でずらすことになってしまいます。

一段組みなら楽勝！という
フォーマット作りを目指す

中級者向けの IT 書でもっとも多いデザインは、「一段組みで文章の途中に図版やソースコードを挟みつつ、章末またはセクション末までリニアに続いていくタイプ」です。このタイプがより速く組めるようになれば、その分を校正期間に回してクォリティアップを図れるようになります。

図 1：一段組みリニア型のイメージ

このタイプのデザイン指定は、たいていの場合、「各パーツのサイズ」と「パーツ間隔」の指定になりますが、それをそのままフォーマットにしてはいけません。
そのまま各パーツがバラバラの状態で組んでしまうと、後でテキストの追加や削除が発生したときに、それ以降を手作業でずらすことになってしまいます。
そのまま各パーツがバラバラの状態で組んでしまうと、後でテキストの追加や削除が発生したときに、それ以降を手作業でずらすことになってしまいます。

一段組みなら楽勝！という
フォーマット作りを目指す

では、組み直しが発生しない完璧な原稿をもらえばいい……というのは組む側の都合であって、作っているものが「解説書」である以上、デザインが複雑だから、DTP が大変だから、間違いがあっても直さないというのは本末転倒です。組みやすいフォーマットを作ることで、テキストや画像追加ぐらいなら楽勝！という状態に持って行きましょう。

1-1 流し込みやすいフォーマットを作る

リード文■□□□□■

一段組みなら楽勝！というフォーマット作りを目指す

中級者向けのIT書でもっとも多いデザインは、「一段組みで文章の途中に図版やソースコードを挟みつつ、章末またはセクション末までリニアに続いていくタイプ」です。このタイプがより速く組めるようになれば、その分を校正期間に回してクォリティアップを図れるようになります。

図1：一段組みリニア型のイメージ

このタイプのデザイン指定は、たいていの場合、「各パーツのサイズ」と「パーツ間隔」の指定になりますが、それをそのままフォーマットにしてはいけません。

1-1 流し込みやすいフォーマットを作る

リード文■□□□□■

一段組みなら楽勝！というフォーマット作りを目指す

中級者向けのIT書でもっとも多いデザインは、「一段組みで文章の途中に図版やソースコードを挟みつつ、章末またはセクション末までリニアに続いていくタイプ」です。このタイプがより速く組めるようになれば、その分を校正期間に回してクォリティアップを図れるようになります。

図1：一段組みリニア型のイメージ

このタイプのデザイン指定は、たいていの場合、「各パーツのサイズ」と「パーツ間隔」の指定になりますが、それをそのままフォーマットにしてはいけません。

ここからソースコードです。

html2indtag.js

```
  //コード以外のHTMLの変換
   var parseHTML = function(src, olmode){
     //img関連。画像全体をfigureタグで囲む
     src = src.replace(/(<img[^>]*>)/g, '<figure>$1</figure>');
     // src = src.replace(/<p><figure>/g, '<figure>');
     // src = src.replace(/<\/figure><\/p>/g, '</figure>');
     //img関連。src=""をhref="file://"に
     src = src.replace(/<img src="([^>]*)>/g, '<img href="file://$1/>');
     //img関連。zoomとclipの指定をimg要素のdata-zoom、data-clip属性に
     src = src.replace(/(<img href="[^\?]*)\?zoom=([0-9]*)/g, '$1" data-
zoom="$2" ');
     src = src.replace(/(data-zoom="[0-9]*" )" /g, '$1');
     src = src.replace(/(<img [^\?>]*)\?clip=([0-9\+]*)"/g, '$1" data-
clip="$2"');
     src = src.replace(/\&amp;clip=([0-9\+]*)"/g, 'data-clip="$1"');
     //キャプションをfigcaptionに
     src = src.replace(/alt="([^"]*)"\/><\/figure>/, '/></figure>\n<figcaption>
<alert>CAP</alert>$1</figcaption>');
     src = src.replace(/alt="([^"]*)"\/><\/p>/, '/></p>\n<figcaption><alert>C
AP</alert>$1</figcaption>');
     //リンクは文字列（url）の形に
     // 誤作動が多すぎるのでナシ
     // src = src.replace(/<a href="([^"]*)">([^<]*)<\/a>/g, '$2 ($1) ');
     //brは改行コードに、hrはとりあえず孤立タグに
     src = src.replace(/<br>/g, '\n');
     src = src.replace(/<hr>/g, '<hr/>');
     //編集コメントや図中文字などの独自拡張（div class="***"で指定しているもの
の処理
     src = src.replace(/<div class="hen">(.*)<\/div>/g,
         '<div_hen><alert>編集</alert>$1</div_hen>');
     src = src.replace(/<div class="zuchuu">(.*)<\/div>/g,
         '<div_zuchuu><alert>図中</alert>$1</div_zuchuu>');
     src = src.replace(/<div class="([^"]*)">(.*)<\/div>/g,
         '<div_$1>$2</div_$1>');
     src = src.replace(/<kbd> [([^] ]*)] <\/kbd>/g, '<kbd>Ì$1Ô</kbd>');
     //その他のspanタグ
     src = src.replace(/<span class="([^"]*)">([^<]*)<\/span>/g,
         '<span_$1>$2</span_$1>');
     //テーブルをスクリプト処理しやすいよう単純化（tbody、theadをカット、thはtdに）
     src = src.replace(/<thead>/, '');
     src = src.replace(/<\/thead>/, '');
     src = src.replace(/<tbody>/, '');
     src = src.replace(/<\/tbody>/, '');
     src = src.replace(/<th/, '<td');
     src = src.replace(/<\/th>/, '</td>');
     //olmodeのときはliをol_liに変える
     if(olmode==true){
       src = src.replace(/<li>([^<]*)<\/li>/, '<ol_li>$1</ol_li>');
     }
     //見出しh1 ～ h6に<alert>を入れる
     if(options.midashialert == true){
       src = src.replace(/<h1[^>]*>/g, '<h1><alert>h1</alert>');
       src = src.replace(/<h2[^>]*>/g, '<h2><alert>h2</alert>');
       src = src.replace(/<h3[^>]*>/g, '<h3><alert>h3</alert>');
       src = src.replace(/<h4[^>]*>/g, '<h4><alert>h4</alert>');
       src = src.replace(/<h5[^>]*>/g, '<h5><alert>h5</alert>');
       src = src.replace(/<h6[^>]*>/g, '<h6><alert>h6</alert>');
     }
     //通常行内のcodeタグを削除
     src = src.replace(/<code>/g, '');
     src = src.replace(/<\/code>/g, '');

     return src;
   };
```

```
//コード以外のHTMLの変換
 var parseHTML = function(src, olmode){
   //img関連。画像全体をfigureタグで囲む
   src = src.replace(/(<img[^>]*>)/g, '<figure>$1</figure>');
   // src = src.replace(/<p><figure>/g, '<figure>');
```


```
//コード以外のHTMLの変換
 var parseHTML = function(src, olmode){
   //img関連。画像全体をfigureタグで囲む
   src = src.replace(/(<img[^>]*>)/g, '<figure>$1</figure>');
   // src = src.replace(/<p><figure>/g, '<figure>');
```

一段組みなら楽勝！という
フォーマット作りを目指す

中級者向けの IT 書でもっとも多いデザインは、「一段組みで文章の途中に図版やソースコードを挟みつつ、章末またはセクション末までリニアに続いていくタイプ」です。このタイプがより速く組めるようになれば、その分を校正期間に回してクォリティアップを図れるようになります。

図 1：一段組みリニア型のイメージ

このタイプのデザイン指定は、たいていの場合、「各パーツのサイズ」と「パーツ間隔」の指定になりますが、それをそのままフォーマットにしてはいけません。
そのまま各パーツがバラバラの状態で組んでしまうと、後でテキストの追加や削除が発生したときに、それ以降を手作業でずらすことになってしまいます。
そのまま各パーツがバラバラの状態で組んでしまうと、後でテキストの追加や削除が発生したときに、それ以降を手作業でずらすことになってしまいます。

□□□□■

□□□□■□□□□■□□□□■□□□□■□□□□■□□□□■□□□□■□□□□■□□□□■
□□□□■□□□□■□□□□■□□□□■□□□□■□□□□■□□□□■□□□□■□□□□■
□□□□■□□□□■□□□□■□□□□■□□□□■□□□□■□□□□■□□□□■□□□□■
□□□□■□□□□■□□□□■□□□□■□□□□■□□□□■□□□□■□□□□■□□□□■
□□□□■□□□□■□□□□■□□□□■□□□□■□□□□■□□□□■□□□□■□□□□■
□□□□■□□□□■□□□□■□□□□■□□□□■□□□□■□□□□■□□□□■□□□□■
□□□□■□□□□■□□□□■□□□□■□□□□■□□□□■□□□□■□□□□■□□□□■
□□□□■□□□□■□□□□■□□□□■□□□□■□□□□■□□□□■□□□□■□□□□■
□□□□■□□□□■

表見出し	表見出し
セル	セル□□□□■□□□□■□□□□■□□□□■□□□□■
セル	セル□□□□■□□□□■□□□□■

1-1 流し込みやすいフォーマットを作る

リード文■□□□□■

一段組みなら楽勝！というフォーマット作りを目指す

中級者向けのIT書でもっとも多いデザインは、「一段組みで文章の途中に図版やソースコードを挟みつつ、章末またはセクション末までリニアに続いていくタイプ」です。このタイプがより速く組めるようになれば、その分を校正期間に回してクォリティアップを図れるようになります。

図1：一段組みリニア型のイメージ

このタイプのデザイン指定は、たいていの場合、「各パーツのサイズ」と「パーツ間隔」の指定になりますが、それをそのままフォーマットにしてはいけません。

1-1 流し込みやすいフォーマットを作る

リード文■□□□□■

一段組みなら楽勝！というフォーマット作りを目指す

中級者向けのIT書でもっとも多いデザインは、「一段組みで文章の途中に図版やソースコードを挟みつつ、章末またはセクション末までリニアに続いていくタイプ」です。このタイプがより速く組めるようになれば、その分を校正期間に回してクォリティアップを図れるようになります。

図1：一段組みリニア型のイメージ

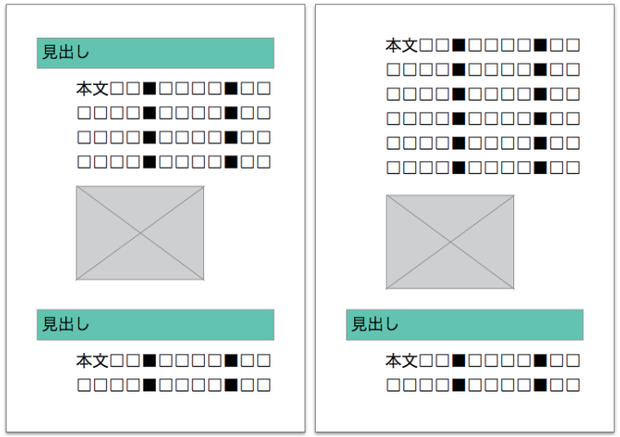
このタイプのデザイン指定は、たいていの場合、「各パーツのサイズ」と「パーツ間隔」の指定になりますが、それをそのままフォーマットにしてはいけません。

1-1 h1 インラインとグリッドを理解する

リード文■□□□□■

h2 一段組みなら楽勝！というフォーマット作りを目指す

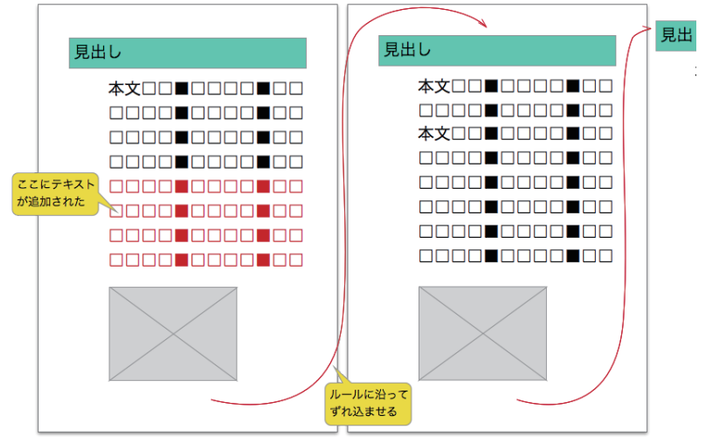
中級者向けのIT書でもっとも多いデザインは、「一段組みで文章の途中に図版やソースコードを挟みつつ、章末またはセクション末までリニアに続いていくタイプ」です。このタイプがより速く組めるようになれば、その分を校正期間に回してクォリティアップを図れるようになります。



CAP 一段組みリニア型のイメージ

このタイプのデザイン指定は、たいていの場合、「各パーツのサイズ」と「パーツ間隔」の指定になりますが、それをそのままフォーマットにしてはいけません。

そのまま各パーツがバラバラの状態で組んでしまうと、後でテキストの追加や削除が発生したときに、それ以降を手作業でずらすことになってしまいます。



CAP デザイン指定そのままのフォーマットでは修正が大変

では、組み直しが発生しない完璧な原稿をもらえばいい……というのは組む側の都合であって、作っているものが「解説書」である以上、デザインが複雑だから、DTPが大変だから、内容に間違いがあっても直さないというのは本末転倒です。組みやすいフォーマットを作ることで、テキストや画像追加ぐらいなら楽勝！という状態に持って行きましょう。

h4 コラム：図解タイプは例外

この文書が対象としているのは、最初に説明したように「一段組みで……リニアに続いていくタイプ」です。見開き（2ページ）単位で完結する図解タイプ（できるシリーズなどの初心者向けの解説書）の本や、凝ったレイアウトの雑誌風の書籍には適用できません。そういうタイプの本は、字数に合わせて書けるベテランライターに執筆してもらうか、編集段階で大幅に変更する許可をもらう必要があるでしょう。

（コラムここまで）

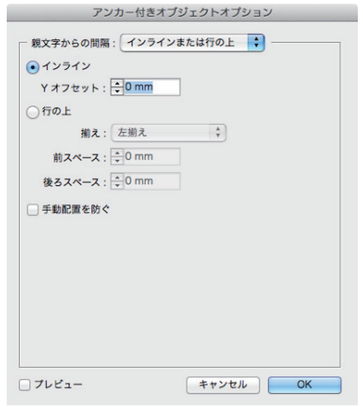
h1 インラインとグリッドを理解する

h2 InDesign のインライン機能を見直す

もともと InDesign は、テキストフレームをリンクさせて複数ページに流し込む機能を持っています。しかし、小説のような文字ばかりの本でしか使えないと思っている人が多いのではないでしょうか？　実は「インライン機能」を使いこなせば、かなり凝ったデザインの本でも、リンクしたテキストフレームを使って組むことができます。インライン機能はテキスト中に他のオブジェクト（フレーム）を挿入する機能ですが、InDesign では「インラインオブジェクト」と「アンカー付きオブジェクト」の２種類に分かれます。

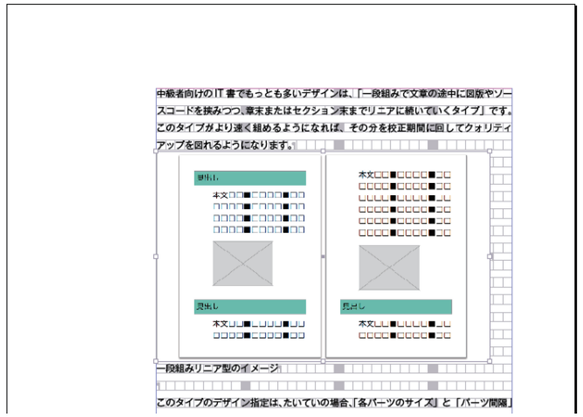
h3 インラインオブジェクト

テキスト中に挿入したフレームを右クリックし、[アンカー付きオブジェクト] →［オプション］を選択すると［アンカー付きオブジェクトオプション］ダイアログボックスが表示されます。
ここで［インラインまたは行の上］の［インライン］を選んだ状態がインラインオブジェクトです。



CAP［インライン］を選択した状態

結果を見るとわかるように、文字とまったく同じ扱いになり、オブジェクトのサイズに応じて周りの文字は追い出されます。ただし、回り込みの設定と違って、複数行のテキストがオブジェクトの横に回り込むことはありません。テキスト中の１文字を大きくした場合と同じ状態になります。

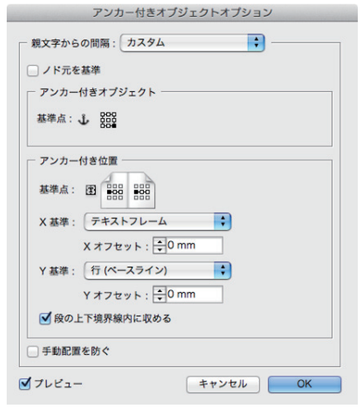


CAP［インライン］を選択した状態

なお、「行の上」という設定もありますが、使ったことがないので割愛します。
インラインにしたときにオブジェクトが他の行に重なってしまう場合は、おそらくインラインにした段落のスタイルで行送りが設定されているはずです。行送りを「自動」にしておけば、オブジェクトのサイズに合わせて行が広がります。

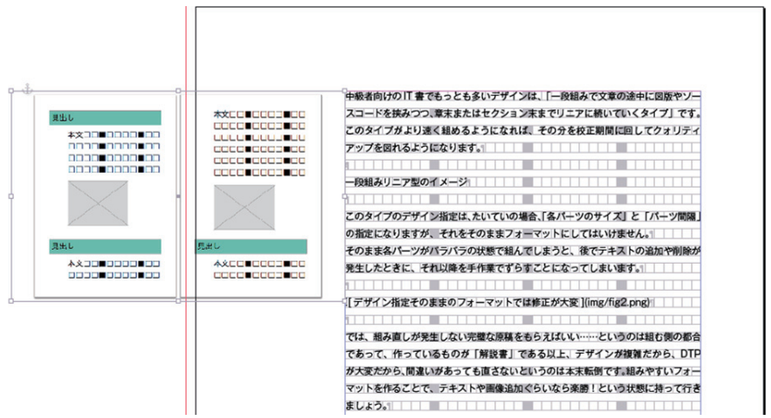
h3 アンカー付きオブジェクト

［アンカー付きオブジェクトオプション］ダイアログボックスで［カスタム］を選んだ状態が、アンカー付きオブジェクトです。インラインよりも設定項目が増え、配置の自由度が上がっています。



CAP［カスタム］を選択した状態

結果を見ると、本文と図版にイカリのマークが付き、本文フレームの外に図版が飛び出しています（配置される場所はオプション設定によって変わります）。テキストには影響ありません。



CAP［カスタム］を選択した状態

２つの設定のうち、メインで使用するのはインラインのほうです。アンカー付きオブジェクトは細かい設定ができますが、その分コントロールが難しくなります。オブジェクトと本文が重ならないようにしたい場合は、回り込み設定を組み合わせなければいけません。また、ページをまたぐ位置に来たときにオブジェクトが見えなくなったり、うっかりアンカー文字を削除してオブジェクトを消してしまったりするトラブルも起きやすくなります。

アンカー付きオブジェクトを使うのは、オブジェクトの横に本文を回り込ませたいときか、セクションタイトルもインライン扱いにしたいときぐらいです。基本的には単純なインラインを使うことにしたほうが、トラブルが少なくなります。

h4 コラム：挿入しただけでアンカー付きオブジェクトになってしまう？

テキストに図版を挿入しただけで、何も設定していないのにアンカー付きオブジェクトになってしまうという現象に悩まされたことはないでしょうか？　その場合は、オブジェクトスタイルなどのデフォルト設定に問題があります。選択ツールを選んで何も選択していない状態にしてから、オブジェクトスタイルや段落スタイルのパネルを確認してみてください。何かのスタイルが選ばれてはいないでしょうか？

InDesign では、未選択状態のスタイルが新たに挿入・新規作成したオブジェクトやテキストに割り当てられるようになっています。未選択状態で各スタイルパネルで「なし」（段落スタイルの場合は「基本段落」）を選んでおけば、初期設定で挿入・新規作成できるようになります。

テキストボックスを作成した時点で予期しない文字スタイルが設定されてしまう場合も原因と対処方法は同じです。未選択状態で各スタイルパネルを確認してください。

（コラムここまで）

コ1 インラインとグリッドを理解する

以下は通常の表のサンプルです。

ファイル・フォルダ名	説明
cssjs	プレビュー用の CSS と JavaScript ファイルを入れておくフォルダ（P. ■■参照）。
doc	Markdown 原稿を入れるフォルダだが、リネームしてもいいし、このフォルダ外に原稿を置いても問題ない。
Gruntfile.js	Grunt の設定ファイル。
indesign-plugins	InDesign 用のプラグイン（P. ■■参照）。インストール後は削除しても問題ない。
kousei-sjis	Just-Right という校正ツールを使用するために、Shift JIS に変換した HTML ファイルを保存しておくフォルダ。削除しても問題ない。
mytemplate.html	プレビュー用 HTML の元になるテンプレートファイル。CSS へのリンクなどを記述しておく（P. ■■参照）。
node_modules	Grunt やプラグインがインストールされているフォルダ。
README.html ～ xml	使用説明ファイル。削除しても問題ない。
sample	テスト用のファイル。削除しても問題ない。

以下はソースコードのサンプルです。

```
module.exports = function(grunt) {
    grunt.initConfig({
        markdown: {
            all: {
                files: [
                    {
                        expand: true,
                        src: ['**/*.md', '!node_modules/**/*.md', '!**/ignore/**/*.md'],
                        ext: '.html'
                    }
                ],
                options: {
                    template: 'mytemplate.html',
                    postCompile: function(src, context) {
                        return src + "<script src='http://localhost:35732/livereload.js'></script>\n";
                    },
                    markdownOptions: {
                        gfm: true,
                        highlight: 'manual'
                    }
                }
            }
        },
        watch:{
            md: {
                files: ['**/*.md', '!**/ignore/**/*.*'],
                tasks: ['markdown'],
            },
            html: {
                files: ['**/*.html', '!kousei-sjis/**/*.html', '!**/ignore/**/*.html'],
                tasks: [ ],
                options: {
                    livereload: 35732
                }
            },
        },
        replace:{
            example: {
                expand: true,
                src: ['**/*.html', '!node_modules/**/*.html', '!kousei-sjis/**/*.html', '!**/ignore/**/*.html', '!**/mytemplate.html'],
                dest: 'kousei-sjis/',
                replacements: [{
                    from:  '<meta charset="utf-8">',
                    to: '<meta charset="sjis">'
                },
                {
                    from: /<script.*<\/script>/g,
                    to:'<!---->'
```

```
                },
                {
                    from: /<link.*>/g,
                    to:'<!---->'
                }
                ]
            }
        },
        utf8tosjis:{
            dist:{
                expand: true,
                src: ['kousei-sjis/**/*.html', '!**/mytemplate.html', '!**/igno
re/**/*.html'],
                overwrite: true,
            }
        },
        html2indtag: {
            dist: {
                expand: true,
                src: ['**/*.html', '!node_modules/**/*.html', '!mytemplate.
html', '!kousei-sjis/**/*.html', '!**/ignore/**/*.html'],
                ext: '.xml'
            },
            options: {
                //見出しが目立ちにくい場合は、trueにすると記号が入る
                midashialert: true
            }
        }
    });

    grunt.loadNpmTasks('grunt-utf8tosjis');
    grunt.loadNpmTasks('grunt-markdown');
    grunt.loadNpmTasks('grunt-contrib-watch');
    grunt.loadNpmTasks('grunt-text-replace');
    grunt.loadNpmTasks('grunt-html2indtag');

    grunt.registerTask("default", ["markdown", "html2indtag", "watch"]);
    grunt.registerTask("kousei", ["markdown", "replace", "utf8tosjis"/*, "wat
ch"*/]);
    grunt.registerTask("xml", ["markdown", "html2indtag"]);
};
```