

BTAP の紹介

BTAP とは？

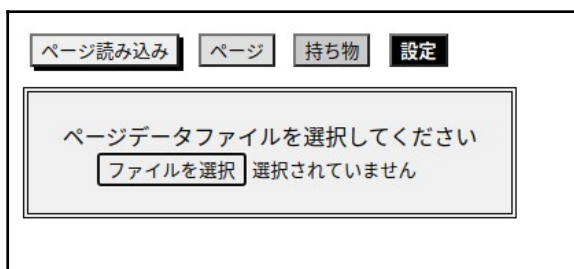
自作の、文字を読むゲームを実装するために作ったツールです。HTMLで作ってあるので、ブラウザで見ることになります。なお、この紹介では、Google Chrome(85.0.4183.102)を用いています。

BTAP 自体には、ゲーム本体のストーリーなどは入っておらず、それが入ったテキストファイルを読み込むことで稼働します。

ここでは便宜上、このストーリーのデータを、「ページデータ」と呼びます。

BTAP の使い方

①ページデータを読み込む



上は BTAP を立ち上げた際の画面です。[ページ読み込み][ページ][持ち物][設定]タブがあります。

ここでは、[ページ読み込み]タブが開いています。(影がついている)
「ファイルを選択」を押し、Story フォルダなどの中に入っているテキストファイルを選択してください。

②[ページ]の表示

ページ読み込み

ページ

持ち物

設定

場所名

ここに、このページの説明文が書いてある。説明文。説明文。説明文。

- 選択肢 1
- 選択肢 2

ファイルの選択に成功したら、[ページ]タブが開きます。選択肢の箇条書きの部分を選ぶことで、ページが進んでいきます。


③[持ち物]の表示

ページ読み込み

ページ

持ち物

設定

 道具1

道具の紹介文がここに来ます

ページを進めていると、道具が手に入ることがあります。持ち物の解説などを[持ち物タブ]で得ることができます。

④各種設定・セーブ

ページ読み込み

ページ

持ち物

設定

色合いを変更

モノクロ▼

セーブデータ書き込み

セーブデータ取得

fig:f

ite:tff

fie:2

セーブデータの読み込み

ファイルを選択

選択されていません

[設定]タブでは、色合いの変更や、セーブデータの書き込み/読み込みができます。

セーブデータもテキストファイルとして扱われており、書き込む際は、実際にテキストファイルに文字をコピーしていただきます。

セーブデータの読み込みは、ページデータと同様にファイルを選択して行います。

ページデータを作る

BTAP のページデータは、次のような形式を取ります。

```
<flag:1>

<item:2>
[0]カギ#どこかのカギ
[1]地図#<img src= 'Story/IMAGE.png' style= 'width:330px; height:220px' />
</item>

<map:3>
[0]
n:場所名
e:ここがこの場所を描写する文章になります
s:選択肢 1#mov(1)
s:選択肢 2#geti(0);geti(1);mov(2)

[1]
n:場所名 2
e:ここがこの場所を描写する文章になります
s:選択肢#mov(0)

[2]
n:終わり
e:これ以上は進めないよ

</map>
<end>
```

フラグ

まず 1 行目には、フラグの設定を行います。

```
<flag:フラグの数>
```

例えば、<flag:3>とすると、内部で 3 つのフラグが用意されます。この時点ではすべて false(フラグが立ってない)ですが、後々、true にしたり(フラグを立てる)、フラグの様子で場合分けができたりします。

持ち物

つぎに、<item:アイテムの個数>~</item>で、持ち物の設定ができます。

```
[番号]名前#説明文
```

番号は 0 から始めていきます。

なお、説明文は innerHTML で表示しているため、上の例のようにすれば、 タグによって画像なども表示できます。

ページ

つづいて<map:総ページ数>~</map>で、各ページの設定ができます。

```
[場所番号]
```

```
n:~
```

```
e:~
```

```
s:~
```

などと書きます。s:は何個あっても構いません。逆に 0 個でも良いです。
n は name(名前)の n、e は explain(説明)の e、s は selection(選択肢)の s の略としています。

s の書き方は、以下の形式です。

```
s:選択肢名#選択したときの処理
```

「#」以後の処理は JavaScript の文法に従います。

コマンド集

- ~ ページに移動する： `mov(~)`
- ~ 番目のフラグを true に： `onflg(~)`
- ~ 番目のフラグを false に： `offlg(~)`
- ~ 番目のアイテムを入手： `geti(~)`
- ~ 番目のアイテムを失う： `losi(~)`
- フラグの情報を知る： `iflg(~)`
- アイテムの有無を知る： `hav(~)`

最後の、`iflg` および `hav` は、JavaScript の `if` 文などと併用できます。そこで、以下のように記述することが可能です。

```
#if(iflg(1)){mov(16);}else if(iflg(0) && hav(0)){mov(14);onflg(1);}else{mov(9);}
```

また、`mov(14);onflg(1);` のように複数の処理を行う場合はセミコロン「`;`」が必要ですが、処理が 1 つの場合、つけてもつけなくても可能です。いずれにせよ、最後尾の処理にはつけなくてもよいと考えますが、どの書き方をとるかは書き手の好みです。

- できる例 `#mov(1)`
- できない例 `#geti(1)mov(1)`
- できる例 `#geti(1);mov(1)`

説明(e:)の改行の省略(v2_0~)

改行を含む文章の場合、「`^`」を行頭に用いることで「`
`」を含まないようにできます。

```
e:a<br>bb<br>ccc
```

は、

```
^ a
^ bb
^ ccc
```

とも記述できます。

また、後ろに改行を含まない文章にする場合は、「^^」を用います。

<pre>^ a ^ bb ^ ^ ^ ccc ^ dddd</pre>	<p>(結果)</p> <pre>a bb cccdddd</pre>
--------------------------------------	--------------------------------------

番号配列(v2_0~)

自由に変更・読み込みができる 12 つの数字が用意されています。初めは、すべて 0 になっています(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)。

配列ですので、要素番号のナンバリングは 0~11 です。

コマンド集

- wnum(n, x) → 第 n 番目の要素に x を書き込む
- rnum(n) → 第 n 番目の要素を得る
- nminn(n1, n2) → 第 n1 番目の要素から、n2 番目の要素の数を引く
- nminx(n, x) → 第 n 番目の要素から x を引く
- naddn(n1, n2) → nminn の足し算版
- naddx(n, x) → nminx の足し算版

例えば、wnum(0, 5); rnum(0); の結果は、5 となります。

wnum(0, 5); wnum(1, 3); nminn(0, 1); rnum(0); rnum(1); の結果は、2、3 となります。

wnum(0, 5); naddx(0, 1); rnum(0); の結果は、6 となります。

Javascript の数学関数の使用(v2_0~)

コマンド集

- rand() → 0~1 までの乱数を作る(Math.random と同じ)
- floor() → 整数化(最も小さい整数を返す)(Math.floor と同じ)
- ceil() → 整数化(最も大きい整数を返す)(Math.ceil と同じ)

span 要素などの組み込み(v2_0~)

例えば、ページ説明(e:)などで、などを含めておくと、あとでその部分を上書きできます。

このときに、

- gebi('id 名', 書き込む内容)

のコマンドを使用します。

(document.getElementById('id 名').innerHTML= 書き込む内容 と同じ)

例えば、ページ 1 の説明(e:)が

ここには匹のがいます。

とした場合、選択(s:)のコマンドを

```
mov(1); wnum(0,3); gebi('xxx', rnum(0)); gebi('yyy', '豚');
```

とすると、

ここには 3 匹の豚がいます。

と直されます。

ページデータの仕上げ

flag、item、map の内容をすべて書いたら、<end>で締めます。

作ったページデータは、適当な名前 (～.txt) で、BTAP/Story 内部に入れてください。また、画像など、このページデータで使うものについても、フォルダにまとめるなどして一緒に BTAP/Story 内部に入れていただくと、あとから取り出しやすいと思います。

なお、flag、item、map で初めに定義するそれぞれの要素の総数が、実際に運用する総数と、じゃっかん違っていても稼働するようです。

文の修飾

map での n:、e:、もしくは item の説明文は、HTML のタグをつけ、修飾することが可能です。代表例を以下に挙げます。

- 改行：

- 太字：～
- ルビ：<r>～#...</r>

ルビについては、`<r>豚#ぶた</r>` とすると、BTAP 内部で

```
<ruby>豚<rp>( </rp><rt>ぶた</rt><rp>)</rp>
```

と直されます。

セーブデータについて

```
flg:ftftf
ite:ftf
fie:10
```

フラグ(fl g)、アイテム(ite)、ページ番号(fie : field の fie)の順に情報が並んでいます。

フラグとアイテムは t:true、f:false、ページ番号はそのまま数字です。

フラグとアイテムは配列で管理されており、上の内容は、

- フラグは[false, true, false, true, false]
- 持ち物の所持は[false, true, false]
- ページ番号は 10

という風に読み込まれます。(データの詳細な説明は省きます)

最後に



BTAP は大変簡素にできているので、
過度な期待をしないでください。