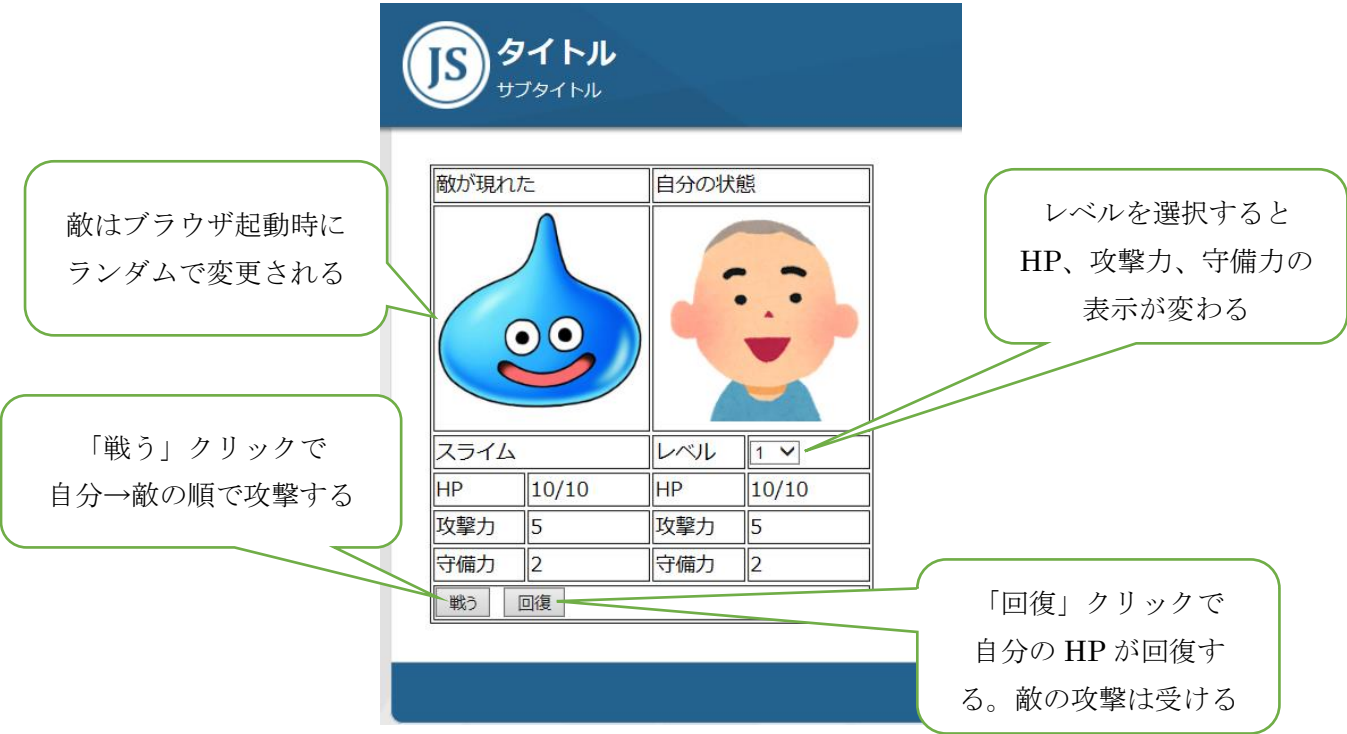


# 問題 モンスターを倒せ3！

以前、作った、モンスターにダメージを与えて倒すプログラムをバージョンアップしよう！



URL : <https://goo.gl/cNsESG>

↑に動くサンプルを置いていますので、動きの確認に使ってください

仕様

イベント	動き
起動する（表示する）	ランダムでモンスターを表示し、モンスターの画像、名前、HP、攻撃力、守備力を表示する。HPは 現在の HP/MAXHP の形式で表示する。 またモンスターの右隣に、自分の画像を表示する。自分の画像の下に、レベル（デフォルト1）、HP、攻撃力、守備力を表示する。 HPは モンスターと同じく 現在の HP/MAXHP の形式で表示する。

レベルを変更する	レベルを変更すると、レベルに応じて HP、攻撃力、守備力が変更される
「戦う」をクリック	<p>自分の攻撃 → 敵の攻撃の順に攻撃をする。</p> <p>攻撃の流れは以下の通り</p> <ul style="list-style-type: none"><li>①ダメージの計算</li><li>②HP の再計算と再表示</li><li>③残り HP チェック</li></ul> <p>0～攻撃力のランダムの数値から相手の守備力を引いたものが相手に与えるダメージになる</p> <p><b>例) 敵の攻撃力が 10、自分の守備力が 5 の場合</b></p> <p>自分が受けるダメージは <b>(0～10 の乱数値) - 5</b> となる。</p> <p>ダメージが 0 以下の場合は 「ダメージを与えられない！」 を表示する</p> <p>受けたダメージを現在の HP から引き、 HP の表示部分を変更する</p> <p>ダメージを受けた結果 HP が 0 以下になった場合は、以下を表示する</p> <p>敵の HP が 0 以下：敵をやっつけた 自分の HP が 0 以下：あなたは死にました</p> <p>敵か自分のどちらかが HP 0 になると、戦うボタンと回復ボタンを押せなくする。 また、「残念」画像を表示する</p>
「回復」をクリック	<p>自分の HP 回復 → 敵の攻撃 の順で処理をする。</p> <p>HP の回復量は、0～50 の乱数値とする。</p> <p>HP は最大値以上にはならない。</p> <p>回復後 HP を再表示する</p>

ひとつ、ひとつ順番に考えていこう！ 以下の順番でやるとわかりやすいと思います。

①初期表示を作る

↓

②攻撃の処理を作る

↓

③回復の処理を作る

↓

④レベルを選択された時の処理を作る

以下、参考にするページなどを紹介

No.	名前	説明
①	初期表示を作る	画像の表示：教科書 P.225 文字の表示：教科書 P.146 オブジェクトからのデータの取得：教科書 P.146
②	攻撃の処理を作る	前回の <b>kadai2</b> を参考にする HP の再表示部分：教科書 P.146、155 など
③	回復の処理を作る	「②攻撃の処理を作る」を元にして、乱数値は敵の HP から引くのではなく、自分の HP に足すようにする。 HP の再表示部分：教科書 P.146、155 など
④	レベルを選択されたときの処理を作る	変更された時のイベント：教科書 P.202

次ページにまた、ヒントがあります。見たい人だけ見て！





### ヒントその1

まずは、初期表示の部分を考えよう！！

`enemyList` は、敵のリストです。オブジェクトの配列になっています（教科書 P.127、141）

```
var enemyList =
[
    { img: "/img/slim.png", name:"スライム", hp:10 , attac: 5, def:2 },
    { img: "/img/ryuou.jpg", name:"竜王", hp:400 , attac: 100, def:30 }
    /*追加する場合はココより下に追加する*/
];
```

上記はオブジェクトが2つあります。エクセルにすると以下のような感じ

img	name	hp	attac	def
	スライム	10	5	2
	竜王	400	100	30

教科書では、オブジェクトが1つでしたが、この例では2つあるので { }

(オブジェクト) の組み合わせが [ ] (配列) の中に2つある

→ オブジェクトが複数ある＝オブジェクトの配列。

配列中の1つめオブジェクトにアクセスするには `enemyList[0]` となる（インデックスは0からはじまる）さらに、オブジェクトにアクセスするにはプロパティ名を使う。

例) `enemyList` の中の「スライム」を取得したい場合

スライムの名前があるオブジェクトが配列の1番目（インデックスは0）にあるので、`enemyList[0]`。

さらに、スライムという名称は `name` というプロパティのところにあるので

`enemyList[0].name`

とすれば、「スライム」を取得できる！

まずは、配列の 1 番目（インデックスは 0）にあるスライムの画像、名前、HP、攻撃力、守備力を表示することを考えよう！

```
document.l("monster_img").src = enemyList[0].
document.l("monster_name").textContent = enemyList[0].
document.l("monster_hp").textContent = enemyList[0].hp+"/"+enemyList[0].hp;
document.l("monster_power").textContent = enemyList[0].
document.l("monster_def").textContent = enemyList[0].
```

今はスライムなので、インデックスは 0 ですが、竜王（配列の 2 番目）の時はインデックスは 1 ですね。

上記が出来たら、ためしに `enemyList[0]` → `enemyList[1]` にして試してみましょう。竜王の情報が表示されるはずです。

では、次に `enemyList[0]` か `enemyList[1]` のどちらか出るか分からないようにする為にはどうしたらいいでしょうか・・・？

そうです、`enemyList` のインデックスを乱数で 0 か 1 どちらが出るか分からない様にしてあげれば、ドキドキ感が増しますね！



では、それを Javascript で書くと・・・

```
var idx = Math.floor(Math.random() * (enemyList.length));

document.l("monster_img").src = enemyList[].
document.l("monster_name").textContent = enemyList[].
document.l("monster_hp").textContent =
enemyList[].hp+"/"+enemyList[].hp;
document.l("monster_power").textContent = enemyList[].
document.l("monster_def").textContent = enemyList[].
```

配列を参照するインデックスを何にすればよいか？の部分は自分で考えてみよう！



## ヒントその2

「攻撃の処理を作る」を作ってみよう

攻撃の処理は「攻撃」ボタンをクリックすることで実行されます。

ですので、まずは「攻撃ボタンをクリック」のイベントを Javascript で作りましょう！

```
////////////////////////////////////
//攻撃ボタンをクリックしたとき
document.("battle").= function() {

    //////////////////////////////////////
    //あなたの攻撃
    var endFlg = attacToEnemy();
    //////////////////////////////////////
    //敵の攻撃
    if( !endFlg ){
        attackedMyself();
    }
}
```

上記イベントのファンクションの中では、自分の攻撃を行うファンクションである `attacToEnemy` と、自分が攻撃をされる `attackedMyself` を呼び出していますね。

`attacToEnemy` と `attackedMyself` の中身はほぼ同じです。今回は `attacToEnemy` のほうを先に作って、`attackedMyself` は後から作ってみます。

```
////////////////////
```

```
//あなたの攻撃
```

```
var attacToEnemy = function(){
```

```
    var endFlg = false;
```

```
    //敵へのダメージを取得
```

```
    var attack = calcDamage(myLvlList[selIdx].hp);
```

与えるダメージ＝

乱数のダメージ － 敵の守備力

enemyList[idx].def は敵の守備力

```
    //敵の体力を減らす
```

```
    var damage = attack - enemyList[idx].def;
```

```
    if( damage <= 0 ){
```

```
        dispMsg("あなたの攻撃：ダメージを与えられない！");
```

```
    }else{
```

```
        quakeImage();
```

画像を揺らすファンクション

```
        now_enemyhp = now_enemyhp - damage;
```

与えたダメージを

敵の体力から引く

```
        //<p>を追加して、「XXX のダメージを与えた」を表示する
```

```
        dispMsg("あなたの攻撃・"+damage+"のダメージを与えた！");
```

```
        document.
```

教科書 P.155

```
        now_enemyhp+"/" + enemyList[idx].hp;
```

```
    }
```

```
    //体力が0になったかのチェック
```

```
    if( now_enemyhp <= 0 ){
```

メッセージの表示。この中で<p>タグを作成

```
        //体力が0になったら、<p>を追加して、「敵をやっつけた」を表示する
```

```
        dispMsg("敵をやっつけた");
```

```
        document.getElementById("battle").disabled = true;
```

```
        document.getElementById("recover").disabled = true;
```

```
        document.getElementById("monster_img").src = "";
```

```
        endFlg = true;
```

```
    }
```

```
    return endFlg;
```

```
}
```

上記の中で太字はファンクションの呼び出し部分です。

ファンクション名	概要
calcDamage	ダメージを計算する。乱数を用い計算したダメージ値を返す
quakeImage	画像を変形させるファンクション
dispMsg	HTML に<p>タグを追加して、引数で指定された文字列を表示する

この中でも dispMsg は HTML を書き換えている部分です。

このファンクションは過去に学んだ知識で作成できるので、作成してみましょう！

```

////////////////////
//メッセージを表示するファンクション
var dispMsg = function(msg){
    var ptag = document.教科書 P.175;

    ptag教科書 P.175 = msg;

    document.getElementById("list").教科書 P.175(ptag);
}

```

ここまで出来たら、attacToEnemy（敵の攻撃）も作りましょう。ほぼ、同じです。

```

////////////////////
//敵の攻撃
var attackedMyself = function(){
    var attack = calcDamage(enemyList[idx].attac);

    var damage = attack - myLvList[selIdx].def;
    if( damage <= 0 ){
        dispMsg("敵の攻撃：ダメージを受けない！");
    }else{
        now_myhp = now_myhp - damage;
        //<p>を追加して、「XXX のダメージを与えた」を表示する
        dispMsg("敵の攻撃："+damage+"のダメージを受けた！");
        document.教科書 P.155 =
                                now_myhp+"/"+myLvList[selIdx].hp;
    }

    //体力が0になったかのチェック
    if( now_myhp <= 0 ){
        //体力が0になったら、<p>を追加して、「あなたは死にました」を表示する
        dispMsg("あなたは死にました");
        document.getElementById("battle").disabled = true;
        document.getElementById("recover").disabled = true;
    }
}

```

ここまでできたら、画面表示→「戦う」ボタンで戦うことが可能なはずです。

確認してみましょう！



## ヒント

### ヒントその3

「回復の処理を作る」を作ってみよう

「攻撃の処理を作る」に似ています。まずは、回復ボタンをクリックされたときを考えましょう。

回復の場合は、「回復」→「敵の攻撃」の順で処理が行われます。

```

////////////////////
//回復ボタンをクリックしたとき
document.教科書 P.155("recover").? = function() {

    //////////////////////
    //回復する
    var recVal = Math.floor(Math.random() * (50));
    now_myhp = ? 体力の回復

    if( now_myhp > myLvList[selIdx].hp ){
        now_myhp = myLvList[selIdx].hp;
    }
    dispMsg("体力が回復"); 体力の表示

    document.教科書 P.155 =
        now_myhp+"/"+myLvList[selIdx].hp;

    //////////////////////
    //敵の攻撃
    ? 敵の攻撃をするファンクションを呼び出す
}

```

ここまでできたら、画面表示→「回復」ボタンで回復することが可能なはずです。

確認してみましょう！



#### ヒントその4

「レベルを選択された時の処理」を作ってみよう

「レベルを選択」→「画面の再表示」の順で処理を行いますので、まずは「レベルの選択」イベントを実装しましょう。

```
document.教科書 P.202("myself_Lv").P.202 = function(){
    selIdx = document.getElementById("myself_Lv").P.203

    document.getElementById("myself_hp").textContent =
        myLvList[?].hp+"/"+myLvList[?].hp;

    document.getElementById("myself_power").textContent =
        myLvList[?].attac;

    document.getElementById("myself_def").textContent =
        myLvList[?].def;
    now_myhp = myLvList[?].hp;
};
```

選択されている項目の  
「value 値」を取得

レベルを取得する部分について、P203 は、form タグについて `getElementById` をしているので、「`document.getElementById('form').select.value`」となっているが、上記で `getElementById` しているのは form タグではなく select ボックスの ID (`myself_Lv`) をしていることに注意しよう。

ここまで出来れば完成です！

出来た人は、モンスターの数を増やしたり、攻撃力の下限値を設定したりなどの改造をしよう！

出来上がったらメールで西野 ([nishino@asojuku.ac.jp](mailto:nishino@asojuku.ac.jp)) までソースを圧縮して送って下さい。