

# モバイル プログラミング 実習2

# 予定

- ネットワーク基礎
  - HTTPメソッド
- HTTP演習
  - 動的なWebページ

# ネットワーク基礎

その4

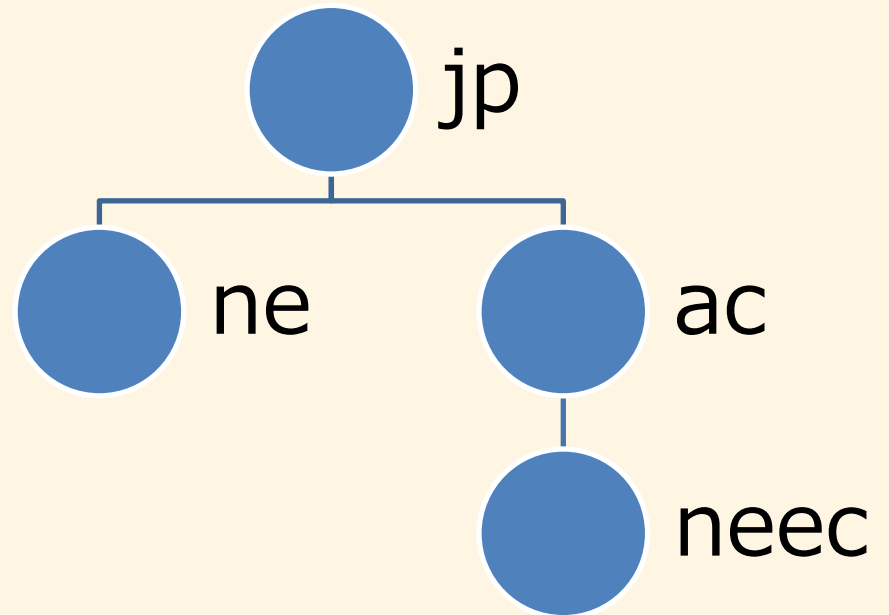
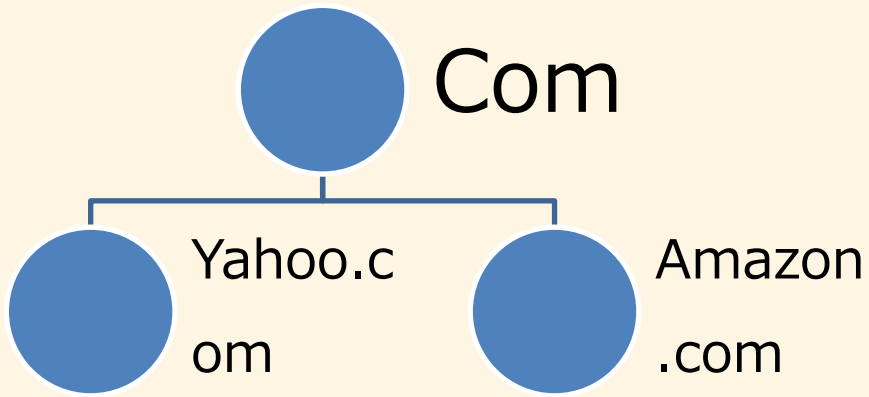
前回の復習

ドメインは  
誰の物？

ドメインは階層構造

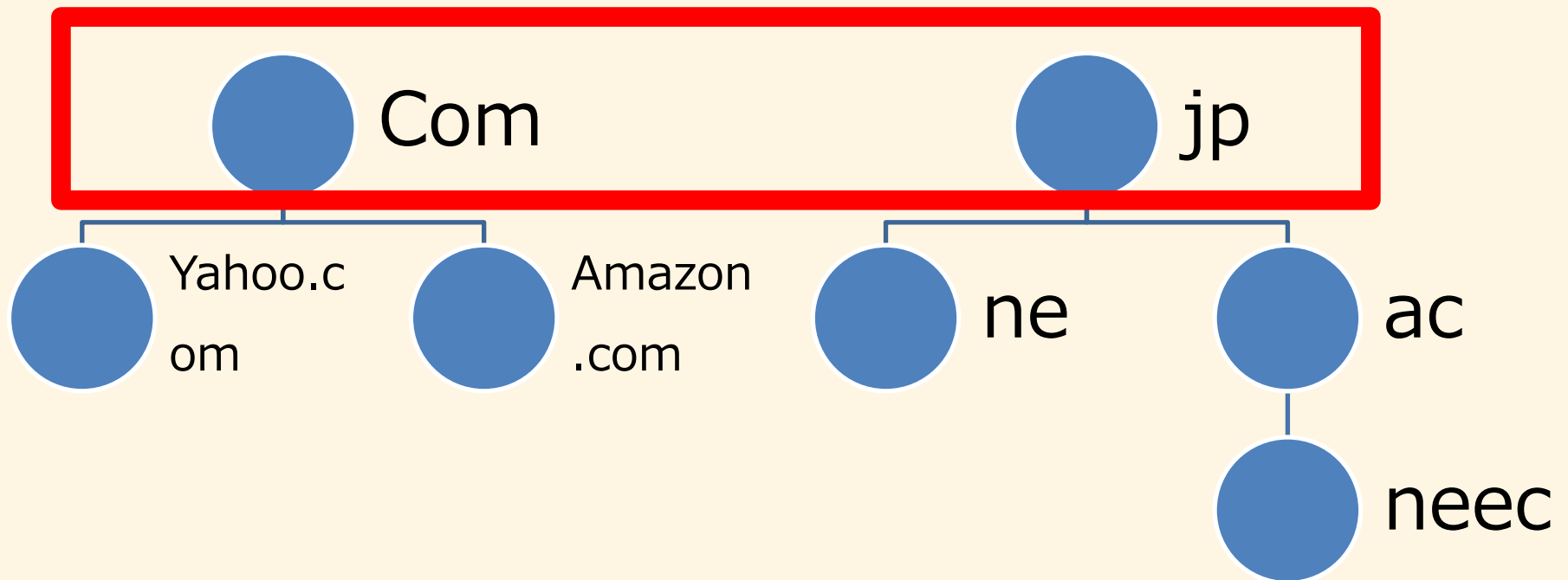
[www.neec.ac.jp](http://www.neec.ac.jp)

# ドメインは階層構造



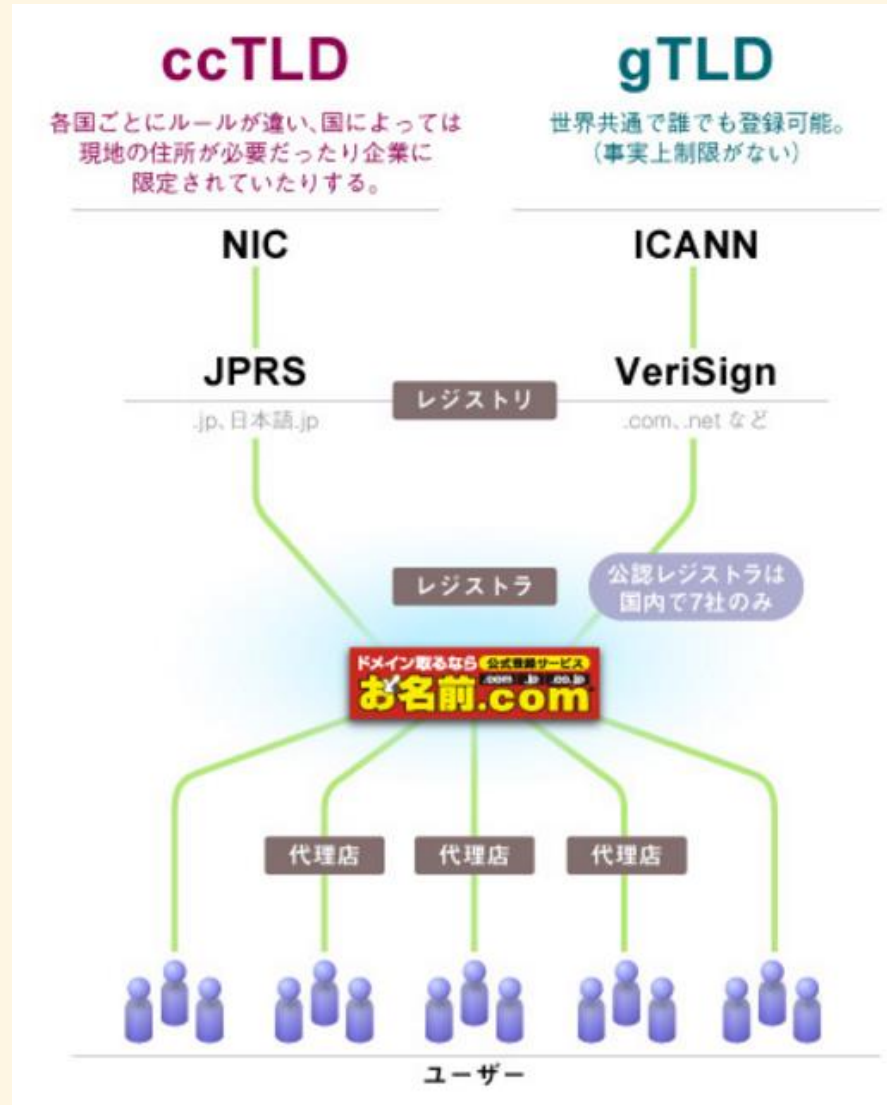
# ドメインは階層構造

## トップレベルドメイン(TLD)





# 「レジストリ」と「レジストラ」



# ドメインは誰でも購入できる

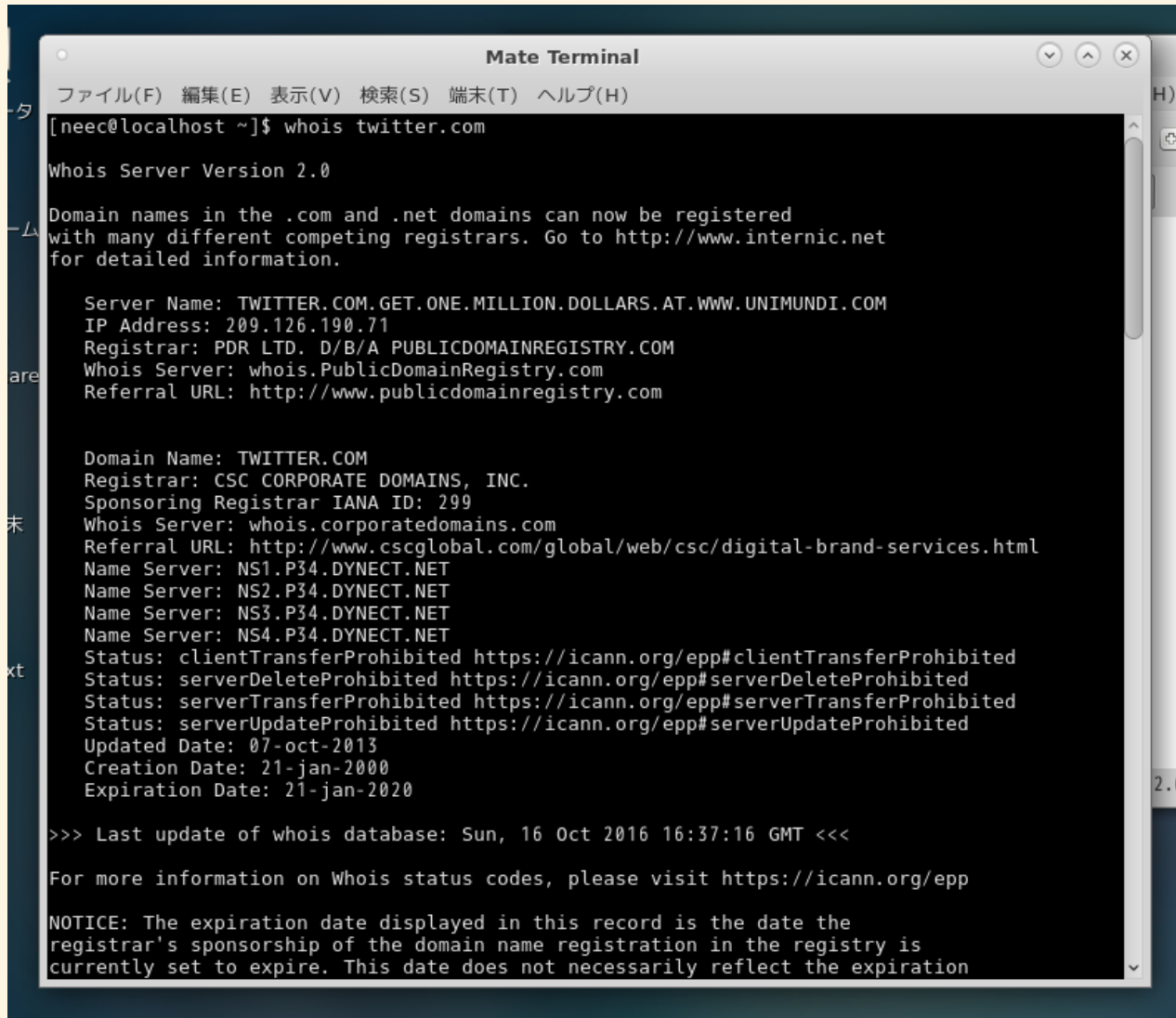


The screenshot shows the MuuMuu Domain website interface. At the top, there's a header with the MuuMuu Domain logo, a blue bear mascot, and navigation links: 取得 (Acquire), 移管 (Transfer), メール (Email), ホームページ (Homepage), 売却 (Sell), お知らせ (Notice), and サポート (Support). Below the header, there's a section titled "ムームードメイン > 価格一覧表" (Muumu Domain > Price List). On the left, there's a sidebar with "各種決済方法" (Various Payment Methods) including "おさいほ! 決済" (Osaiho! Payment), "クレジットカード決済" (Credit Card Payment), "コンビニ決済" (Convenience Store Payment), and "銀行振込" (Bank Transfer). It also lists "ご利用可能なコンビニ" (Convenience Stores you can use) with logos for Lawson, FamilyMart, and Seicomart, and "ご利用可能なクレジット" (Credit cards you can use) with logos for Visa, MasterCard, and American Express. At the bottom left, there's a logo for "Osaiho! ネットのおサイフ おさいほ!" (Osaiho! Net's Osaiho!). On the right, there's a table titled "価格一覧表" (Price List) with columns for "ドメイン" (Domain), "取得" (Acquire), "更新" (Renew), and "移管" (Transfer). The table lists various domains and their corresponding prices in Japanese Yen (¥).

ドメイン	取得	更新	移管
com	¥499	¥1,280	¥1,280
net	¥699	¥1,280	¥1,280
日本語.xyz	¥99	¥1,480	¥1,480
info	¥299	¥1,480	¥1,480
日本語.com	¥499	¥1,280	¥1,280
xyz	¥99	¥1,480	¥1,480
co.jp ※1	¥2,480	¥3,780	—
biz	¥399	¥1,480	¥1,480
日本語.net	¥699	¥1,280	¥1,280
日本語.biz	¥399	¥1,480	¥1,480
top	¥99	¥2,980	—

<https://muumuu-domain.com/?mode=price>

# ドメインの所有者を調べるのは 「whois」 コマンド

A screenshot of a terminal window titled "Mate Terminal". The window has a menu bar with options: ファイル(F), 編集(E), 表示(V), 検索(S), 端末(T), ヘルプ(H). The terminal shows the command `[nec@localhost ~]$ whois twitter.com` and its output. The output includes information about the whois server and the domain registration details for twitter.com.

```
Mate Terminal
ファイル(F) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
[nec@localhost ~]$ whois twitter.com

Whois Server Version 2.0

Domain names in the .com and .net domains can now be registered
with many different competing registrars. Go to http://www.internic.net
for detailed information.

Server Name: TWITTER.COM.GET.ONE.MILLION.DOLLARS.AT.WWW.UNIMUNDI.COM
IP Address: 209.126.190.71
Registrar: PDR LTD. D/B/A PUBLICDOMAINREGISTRY.COM
Whois Server: whois.PublicDomainRegistry.com
Referral URL: http://www.publicdomainregistry.com

Domain Name: TWITTER.COM
Registrar: CSC CORPORATE DOMAINS, INC.
Sponsoring Registrar IANA ID: 299
Whois Server: whois.corporatedomains.com
Referral URL: http://www.cscglobal.com/global/web/csc/digital-brand-services.html
Name Server: NS1.P34.DYNECT.NET
Name Server: NS2.P34.DYNECT.NET
Name Server: NS3.P34.DYNECT.NET
Name Server: NS4.P34.DYNECT.NET
Status: clientTransferProhibited https://icann.org/epp#clientTransferProhibited
Status: serverDeleteProhibited https://icann.org/epp#serverDeleteProhibited
Status: serverTransferProhibited https://icann.org/epp#serverTransferProhibited
Status: serverUpdateProhibited https://icann.org/epp#serverUpdateProhibited
Updated Date: 07-oct-2013
Creation Date: 21-jan-2000
Expiration Date: 21-jan-2020

>>> Last update of whois database: Sun, 16 Oct 2016 16:37:16 GMT <<<

For more information on Whois status codes, please visit https://icann.org/epp

NOTICE: The expiration date displayed in this record is the date the
registrar's sponsorship of the domain name registration in the registry is
currently set to expire. This date does not necessarily reflect the expiration
```

# ドメインは誰の物？

- TLD毎に管理者がいる
  - 管理者のことを「レジストリ」
  - 登録者のことを「レジストラ」
- ドメインは誰でも購入することが可能。

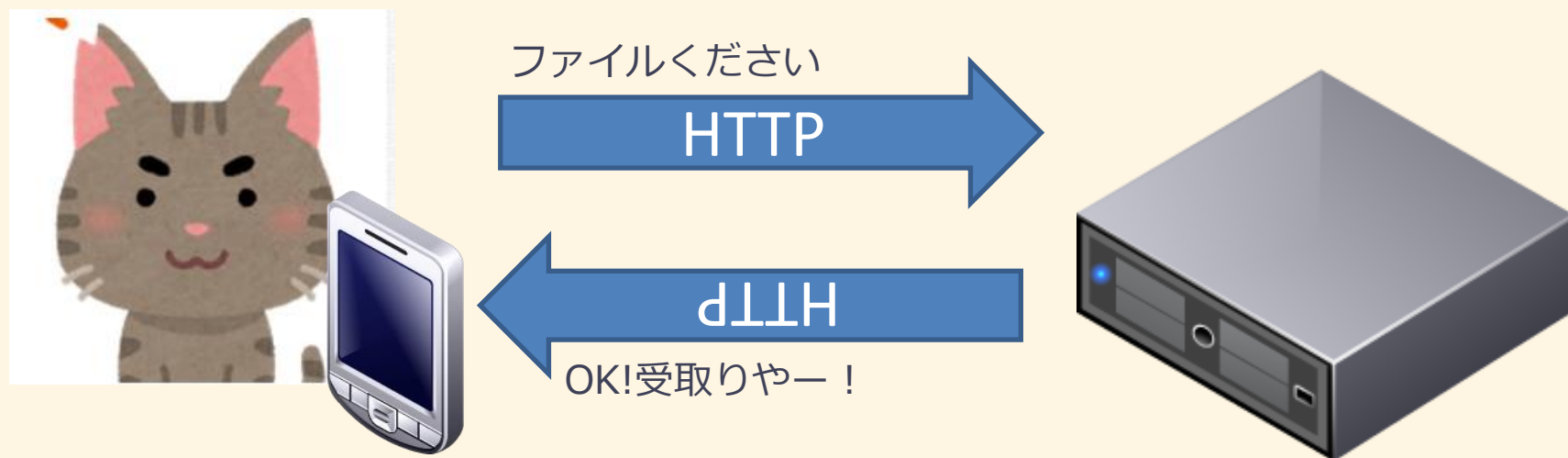
# HTTP メソッド

今日の

このコーナーは

伏線回収

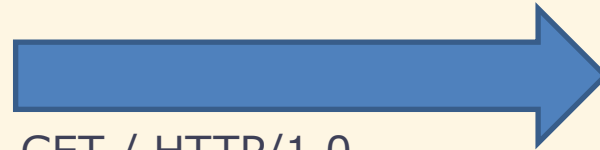
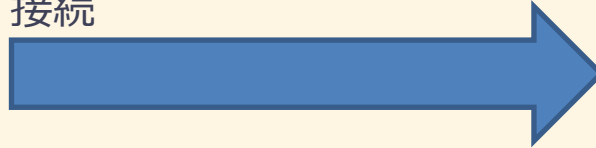
# プロトコルとは何か？



# HTTP



接続



GET / HTTP/1.0



HTTP/1.1 200 OK

Server: nginx

Date: Sun, 09 Oct 2016

13:19:36 GMT

Content-Type: text/html;  
charset=UTF-8

Connection: close



# telnet で http を手打ちする

クライアントから入力

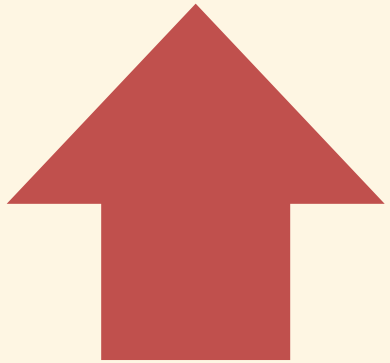
```
$ telnet www.yahoo.co.jp 80  
GET / HTTP/1.0
```

サーバからの返却

```
HTTP/1.1 200 OK  
Server: nginx  
Date: Sun, 09 Oct 2016 13:19:36 GMT  
Content-Type: text/html; charset=UTF-8  
Connection: close  
(以降HTML)
```

# メソッド

```
$ telnet www.yahoo.co.jp 80  
GET / HTTP/1.0
```



この部分が「メソッド」

# メソッド

- メソッドは「**コマンド**」のような物
- 例えば
  - **GET / POST**
    - ファイルをください
  - **HEAD**
    - ファイルの情報をください。  
(中身はいらないです)

# メソッド

- その他にも

- **PUT**

- こちら(クライアント)からファイルを送信するのでサーバに保存してください。

- **DELETE**

- サーバ上のファイルを削除してください。

※他にもありますが、ひとまずこの5つを覚えておけば良いです。

# GETとPOSTの違い

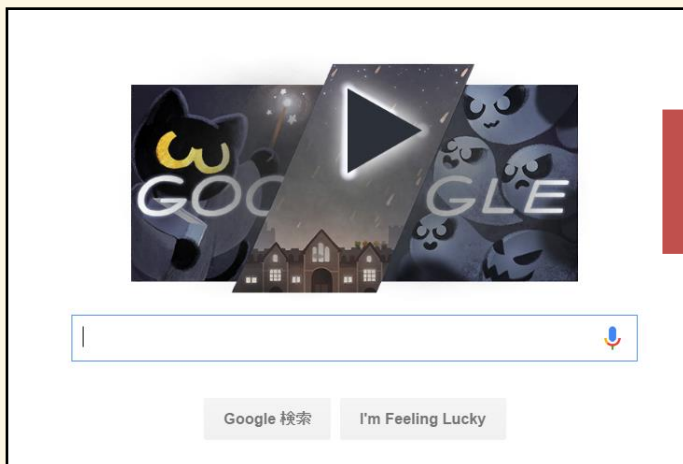
# GET/POSTの使い分け

- GETとPOSTを使い分ける必要が出てくるのは、**サーバに値を渡すとき**です。

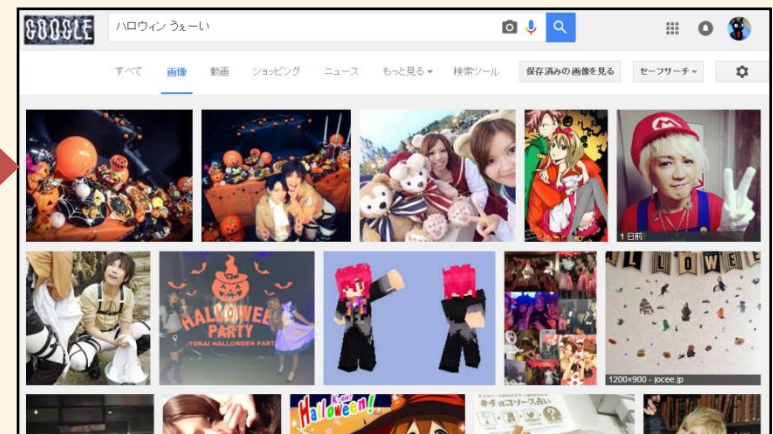
# GET/POSTの使い分け

- GETとPOSTを使い分ける必要が出てくるのは、**サーバに値を渡すとき**です。

入力

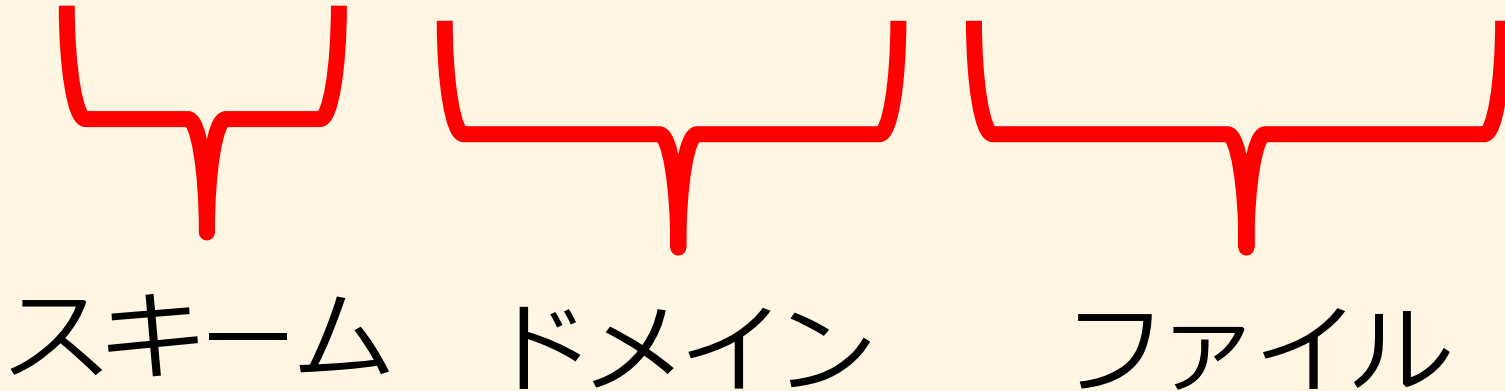


入力した値によって  
結果を変えたい



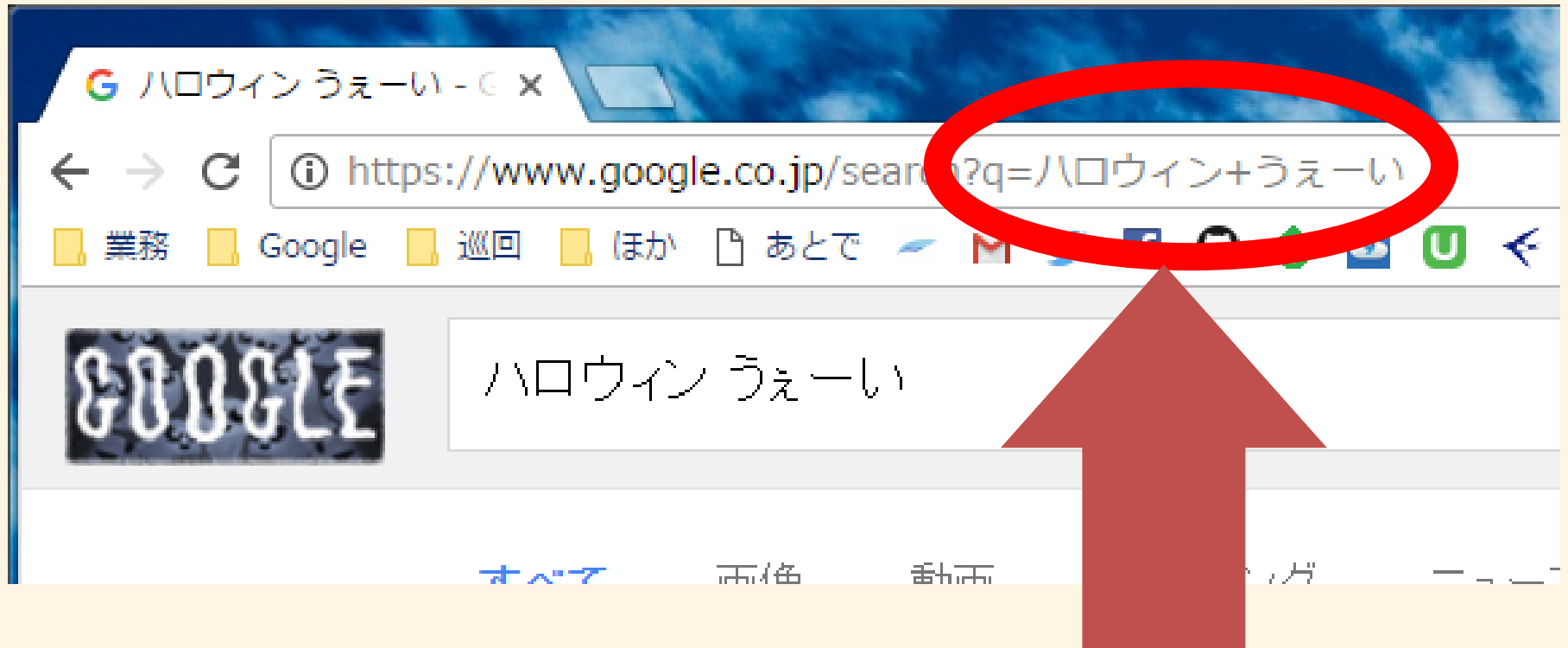
# URLの意味

<https://twitter.com/GachapinBlog>





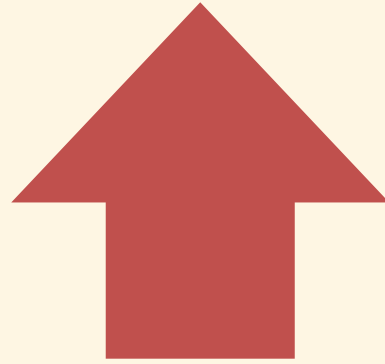
# URLにクエリーを追加



この部分が値。  
「クエリー」と呼びます。

手打ちする際も追加OK

```
$ telnet www.google.co.jp 80  
GET /search?q=cat HTTP/1.0
```



telnetを利用した場合も  
同様に記述できます。

# GETの問題点

- **文字列長の制限**

- HTTPの仕様上は制限ない。ブラウザの実装による問題。
- 概ね2000byte以内。

- **セキュリティの問題**

- URLはWebサーバのアクセスログなどに保存されます。
- 例えば**個人情報**や**パスワード**などがクエリーにあると一緒に記録されてしまう。

# POSTを手打ちする

```
$ telnet www.example.com 80  
POST /receive.php HTTP/1.0  
Content-Length: 1234  
  
q=cat&foo=bar&hoge=huga...
```

※クエリーの書式はこの後説明を行います。

# POSTを手打ちする

```
$ telnet www.example.com 80  
POST/receive.php HTTP/1.0  
Content-Length: 1234  
  
q=cat&foo=bar&hoge=huga...
```

※クエリーの書式はこの後説明を行います。

# POSTを手打ちする

```
$ telnet www.example.com 80  
POST /receive.php HTTP/1.0  
Content-Length: 1234  
  
q=cat&foo=bar&hoge=huga...
```

※クエリーの書式はこの後説明を行います。

# POSTを手打ちする

```
$ telnet www.example.com 80  
POST /receive.php HTTP/1.0  
Content-Length: 1234
```

q=cat&foo=bar&hoge=huga...

※クエリーの書式はこの後説明を行います。

# GET

- **利点**

- ブラウザのURL入力欄に指定できる
  - A要素(リンクタグ)での指定も可能
- ブラウザのキャッシュが効く

- **制約**

- 文字列長の制限
- セキュリティ



# POST

## • 利点

- 大きなサイズのデータが送信できる
- セキュリティ上安心

※いずれもGETと比較した場合。

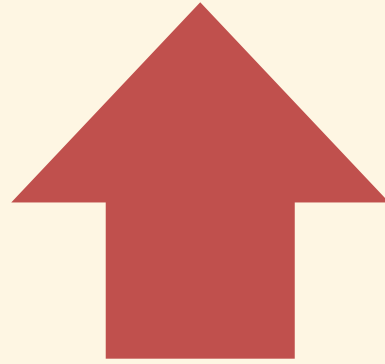
## • 制約

- ブラウザの入力欄で指定できない
- キャッシュが効かない
  - 基本的に毎回送信する必要がある

# クエリーの書式

手打ちする際も追加OK

```
$ telnet www.google.co.jp 80  
GET /search?q=cat HTTP/1.0
```



telnetを利用した場合も  
同様に記述できます。

# POSTを手打ちする

```
$ telnet www.example.com 80  
POST /receive.php HTTP/1.0  
Content-Length: 1234
```

q=cat&foo=bar&hoge=huga...

# クエリーの書式

クエリーが一つ

name=value

クエリーが複数

q1=value1&q2=value2

q1=value1&q2=value2&q3=value3

&で結合することでいくらでも指定できます

# URLエンコード

- 以下の値の利用には**エンコード**が必要
  - URLで使用できない文字
  - マルチバイト文字
  - バイナリ

※ここでは超ざっくり説明しています。

# URLエンコード

and those reserved characters not acting as delimiters, define each component's identifying data.

## 2.1. Percent-Encoding

A percent-encoding mechanism is used to represent a data octet in a component when that octet's corresponding character is outside the allowed set or is being used as a delimiter of, or within, the component. A percent-encoded octet is encoded as a character triplet, consisting of the percent character "%" followed by the two hexadecimal digits representing that octet's numeric value. For example, "%20" is the percent-encoding for the binary octet "00100000" (ABNF: %x20), which in US-ASCII corresponds to the space character (SP). [Section 2.4](#) describes when percent-encoding and decoding is applied.

pct-encoded = "%" HEXDIG HEXDIG

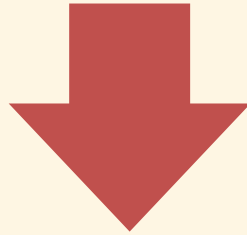
The uppercase hexadecimal digits 'A' through 'F' are equivalent to the lowercase digits 'a' through 'f', respectively. If two URIs differ only in the case of hexadecimal digits used in percent-encoded octets, they are equivalent. For consistency, URI producers and normalizers should use uppercase hexadecimal digits for all percent-encodings.

## 2.2. Reserved Characters

URIs include components and subcomponents that are delimited by characters in the "reserved" set. These characters are called "reserved" because they may (or may not) be defined as delimiters by the generic syntax, by each scheme-specific syntax, or by the

# URLエンコード

肉



%E8%82%89



# URLエンコード

name=%E8%82%89

※name部分にマルチバイト文字を使った場合も、同様にエンコードが必要です。  
※なるべく左辺には使わないようにするのがおすすめです。

# URLエンコード



## urlencode

(PHP 4, PHP 5, PHP 7)

urlencode — 文字列を URL エンコードする

### 説明

```
string urlencode ( string $str )
```

この関数は、URL の問い合わせ部分に使用する文

<http://php.net/manual/ja/function.urlencode.php>

[https://developer.mozilla.org/ja/docs/Web/JavaScript/Reference/Global\\_Objects/encodeURIComponent](https://developer.mozilla.org/ja/docs/Web/JavaScript/Reference/Global_Objects/encodeURIComponent)

# URLエンコード

- URL中に使用できない文字やバイナリはあらかじめエンコードする必要がある。
- URLエンコード/デコードを行う手段が提供されている言語では、それらを用いる。

# HTTP演習

## その4

前回の復習

# JavaScript入門

# 目次 その1

- HelloWorld
  - alert, 定数, 変数, 配列, for
  - 関数, クロージャ
- JSを記述する場所
  - head, body
- 演習
  - FizzBuzz

# 目次 その2

- デベロッパーツール
  - Consoleにfizzbuzz
- idとclass
  - 要素を追加する(DOM)
    - 一箇所だけ
    - 複数同時に

※プロフィールページをいじります。  
(プロフィール項目を追加する)



# 目次 その3

- イベント
  - フォーム要素(HTML)
  - 今回は古い書き方
    - onclick=""

# 動的な Webページ



# 演習サマリー1

- HTML フォーム
  - テキストボックス、テキストエリア
  - プルダウン
  - ラジオ、チェックボックス
- 値の受取り方
  - `$_GET`, `$_POST`, `$_REQUEST`

# 演習1. MAP表示

配列

```
$map= [  
    [0, 0, 0, 0]  
    , [0, 1, 2, 0]  
    , [0, 2, 1, 0]  
    , [0, 0, 0, 0]  
];
```



実行結果

```
$ php map.php  
_ _ _ _  
_ A B _  
_ B A _  
_ _ _ _
```

これをWebページ風に表示してることにしましょう

# 演習1. MAP表示

配列

```
$map= [  
    [0, 0, 0, 0]  
    , [0, 1, 0, 0]  
    , [0, 0, 0, 0]  
    , [2, 2, 2, 2]  
];
```



0

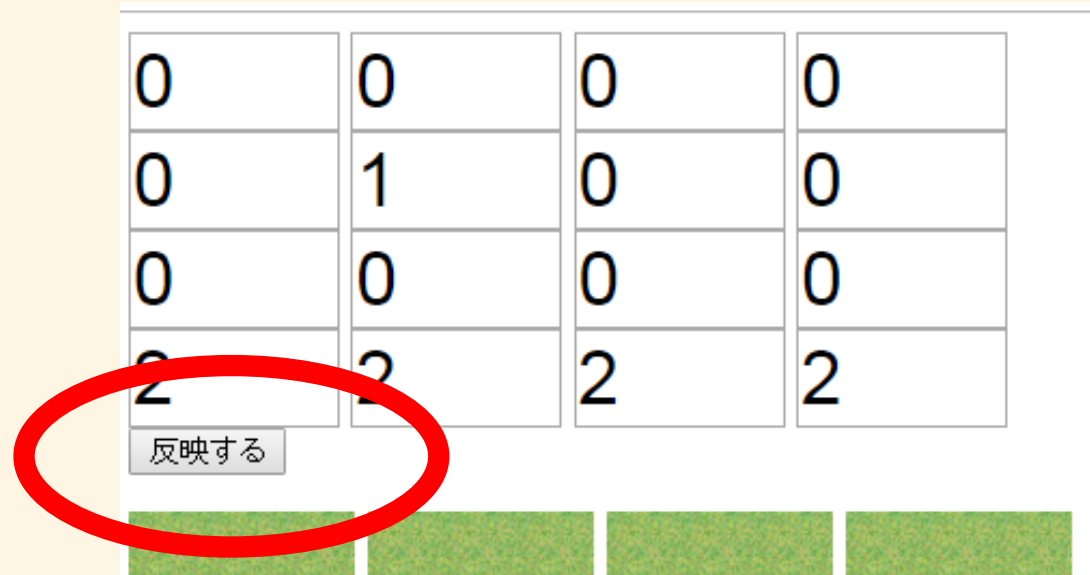
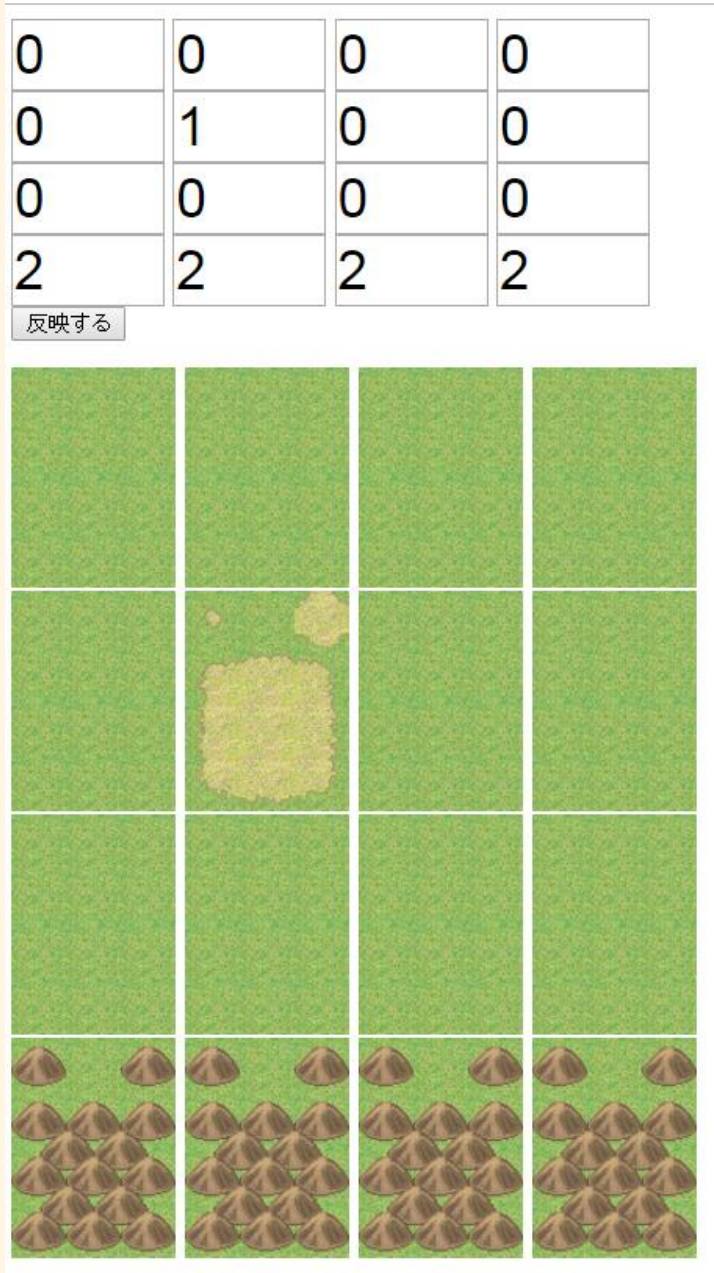


1



2

# 演習2. MAPエディタ

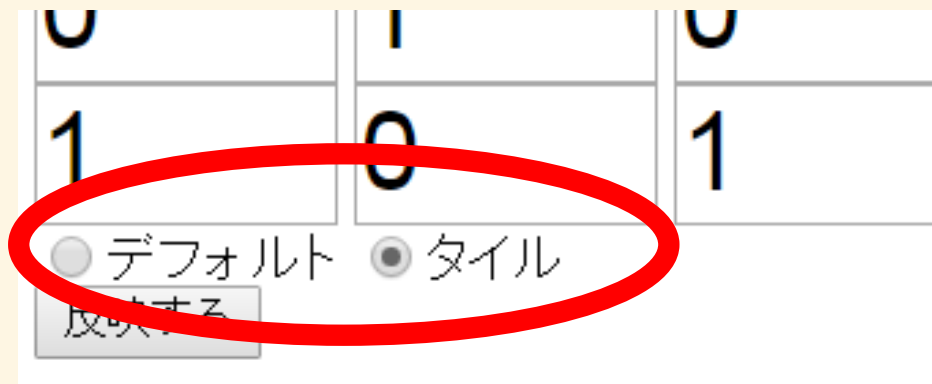
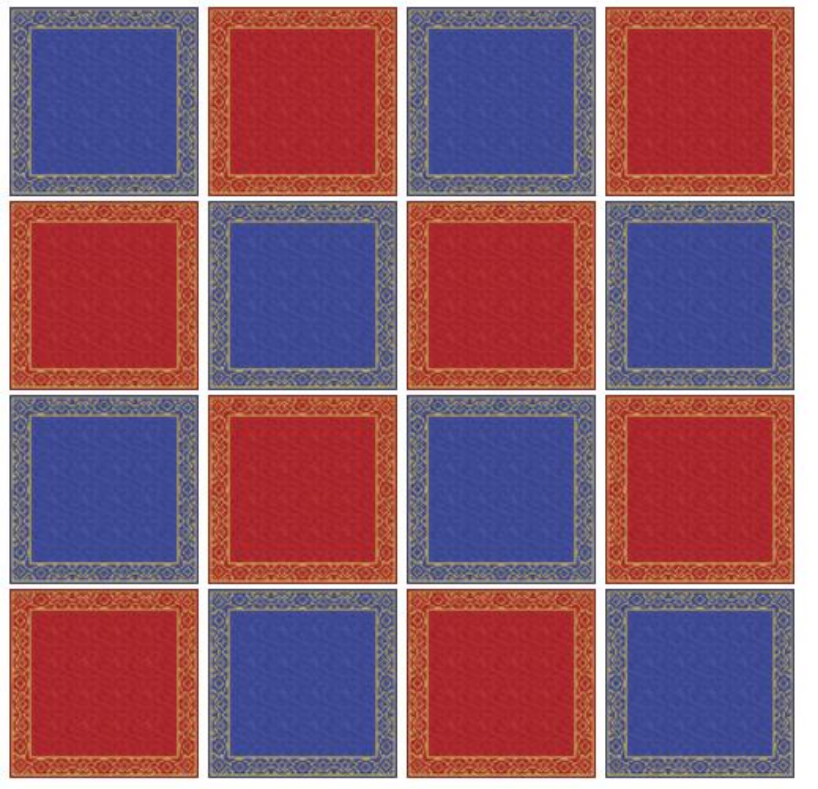


ボタンを押すと  
MAPに反映

# 演習3. MAPエディタ改

0	1	0	1
1	0	1	0
0	1	0	1
1	0	1	0

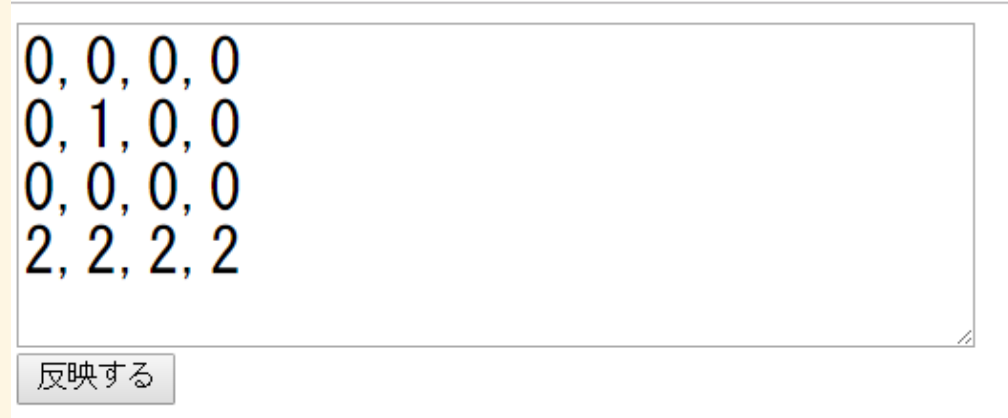
☐ デフォルト ☒ タイル  
反映する



☐ デフォルト ☒ タイル  
反映する

ラジオボタンで  
スキンを切り替え

# 演習4. MAPエディタ改2



入力欄をテキスト  
エリアに変更