

TeraTerm で Telnet

Seiichi Nukayama

2021-08-22

目次

1	XAMPP のインストール	1
2	TeraTerm のインストール	1
2.1	TeraTerm のダウンロード	1
3	Web サーバーを動かす	9
3.1	作業フォルダで HTML ファイルを作成	9
3.2	簡易 Web サーバーを起動	9
4	Telnet で Web サーバーと通信する	10
4.1	Telnet とは	10
4.2	TeraTerm の設定	10
5	Telnet で GET リクエスト	13
5.1	ブラウザは何をやっているのか?	13
5.2	GET リクエスト	13
5.3	TeraTerm で Get リクエスト	13
5.4	さまざまな GET リクエスト	16
5.5	ブラウザのやっていること	17
6	Telnet で POST リクエスト	18
6.1	サーバー側で準備をする	18
6.2	Telnet での POST 送信のしかた	21
6.3	TeraTerm で POST 送信をやってみる	23
7	PHP と JavaScript	28
7.1	PHP のはたらき	28
7.2	JavaScript のはたらき	28
7.3	TeraTerm でやってみる	30

Web ブラウザで Web サイトを閲覧するというのは、いったい何をしているのだろうか。

ブラウザと Web サーバーでどういうやりとりをしているのだろうか。

それを Telnet 接続によって体験してみる。

そのために PHP と TeraTerm が必要である。PHP は、その簡易サーバー機能を使う。また、簡単な php スクリプトも使う。

TeraTerm は、それを使って サーバーに Telnet 接続をする。

1 XAMPP のインストール

別文書”XAMPP のインストールと設定”を参照。

2 TeraTerm のインストール

2.1 TeraTerm のダウンロード

Google 検索「teraterm」とすると、以下のサイトが一番上にくるはず。



このリンクをクリックすると、以下のページになる。

プロジェクトの説明



画像一覧

Tera Term は、オリジナルの [Tera Term Pro 2.3](#) の原作者公認の後継版です。オープンソースで開発されており、UTF-8 表示に対応しています。また、SSH1 対応モジュール TTSSH を拡張し、SSH2 プロトコルをサポートしています。

インストール

ダウンロードが完了したら、パッケージをクリック（もしくはダブルクリック）して実行する。するとインストールウィザードが起動し、従ってインストールする。なお、途中でインストール...

使い方

デスクトップに作成されたTera Termのショートカットアイコンをクリック（あるいはダブルクリック）して起動するとメインウィンドウが表示される。メインウィンドウと「新しい接続」...

ダウンロード

最新リリース

[Tera Term 4.106](#) (日付: 2021-06-05)

[Tera Term RC 4.106 RC](#) (日付: 2021-05-22)

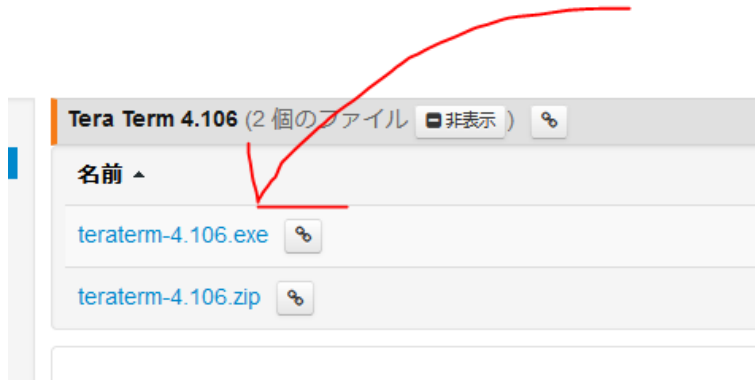
[Tera Term 4.105](#) (日付: 2019-12-07)

[Tera Term RC 4.105 RC](#) (日付: 2019-11-23)

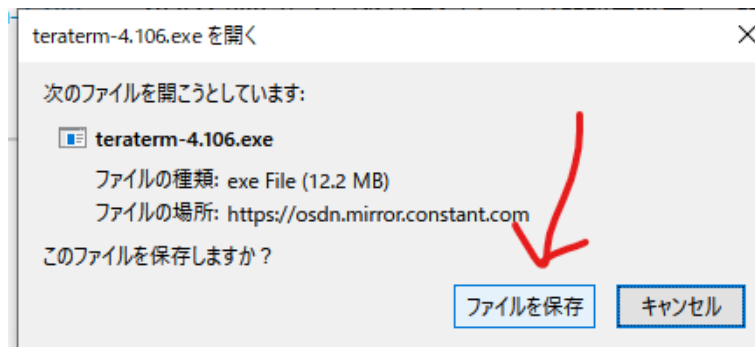
[Tera Term 4.104](#) (日付: 2019-08-31)

”ダウンロード”の項目の”最新リリース”から、Tera Term 4.106 をクリック。

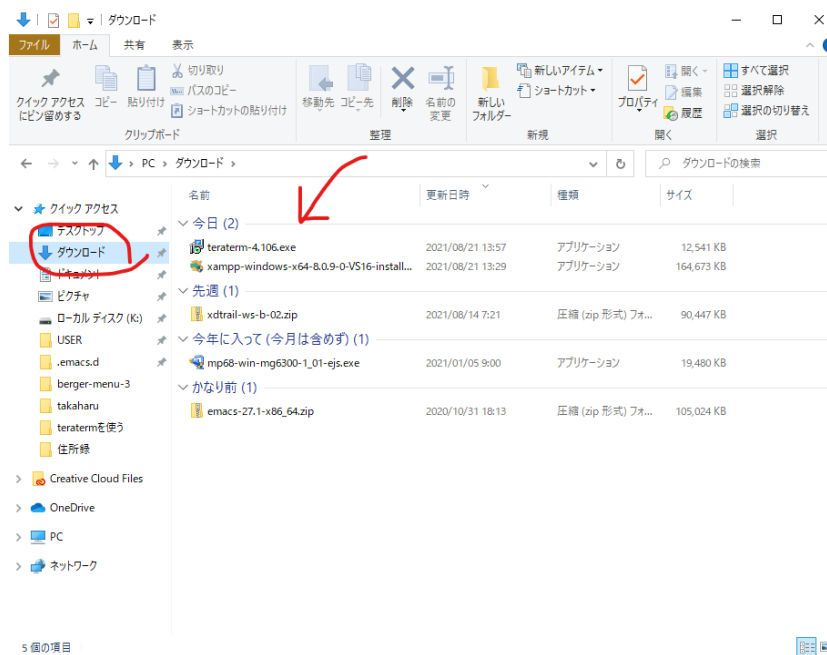
次に開いたページで、teraterm-4.106.exe と teraterm-4.106.zip の 2 つが選択できるが、どちらも内容は同じ。ここでは、teraterm-4.106.exe を選択する。



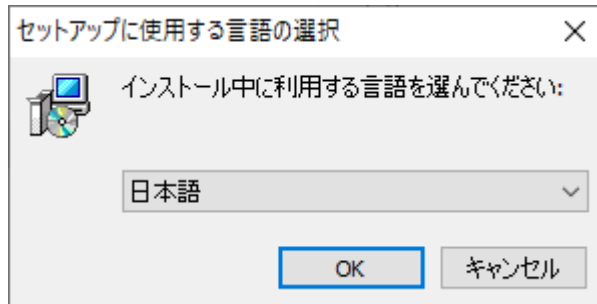
次のダイアログが表示されるので、”ファイルを保存” とする。



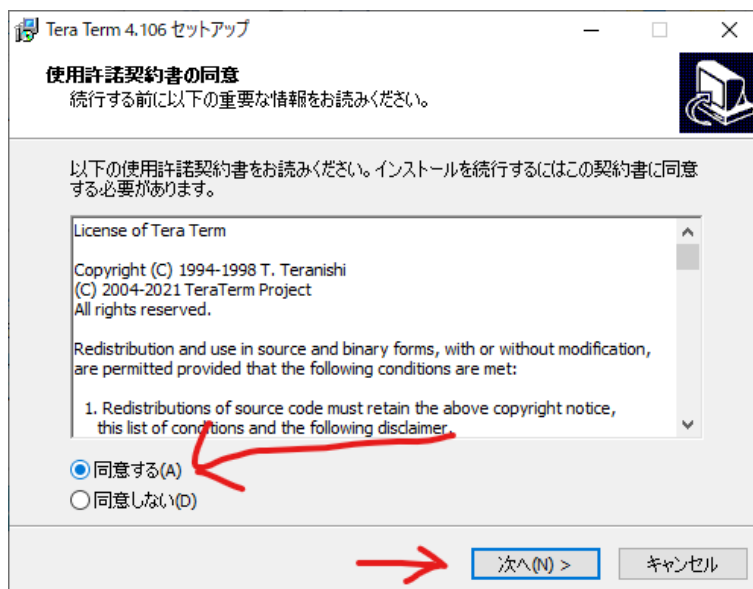
すぐに保存される。ダウンロードフォルダに保存されるので、ダウンロードフォルダを開く。



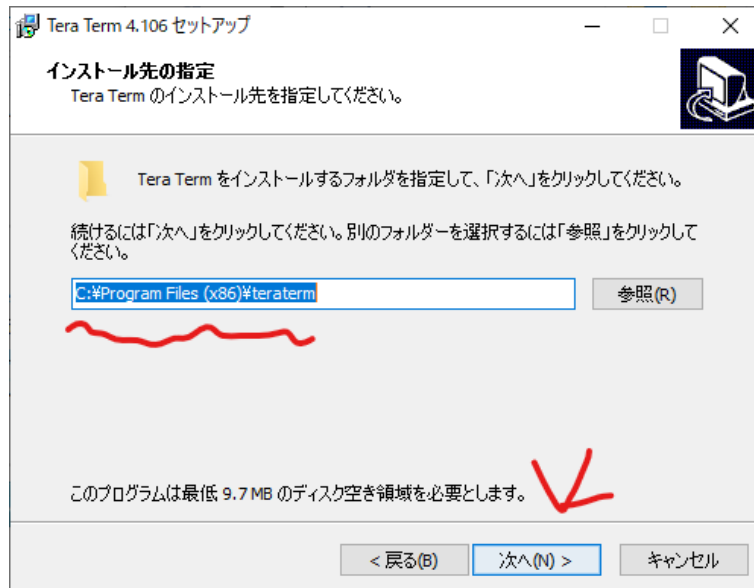
ダウンロードされた teraterm-4.106.exe をダブルクリックして、インストールを始める。
インストールする言語に「日本語」を選択。



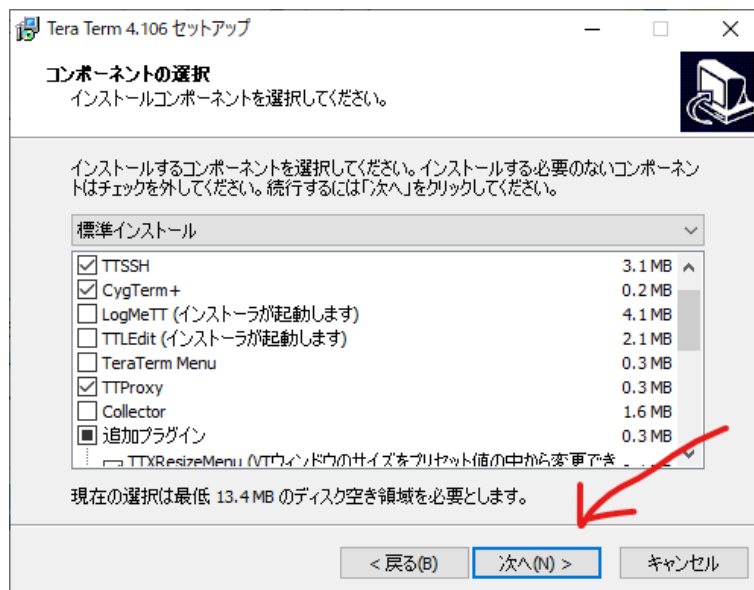
「使用許諾契約書の同意」は「同意する」にチェック。



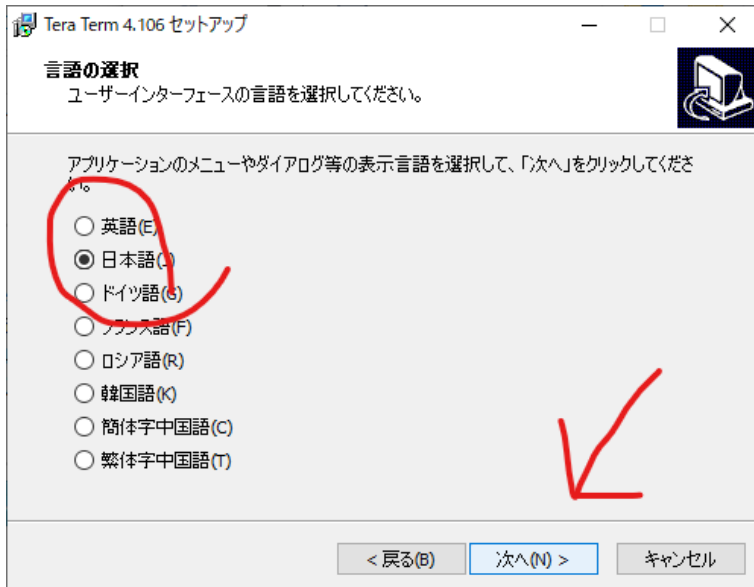
インストール先を確認。(C:\Program Files (x86)\teraterm)



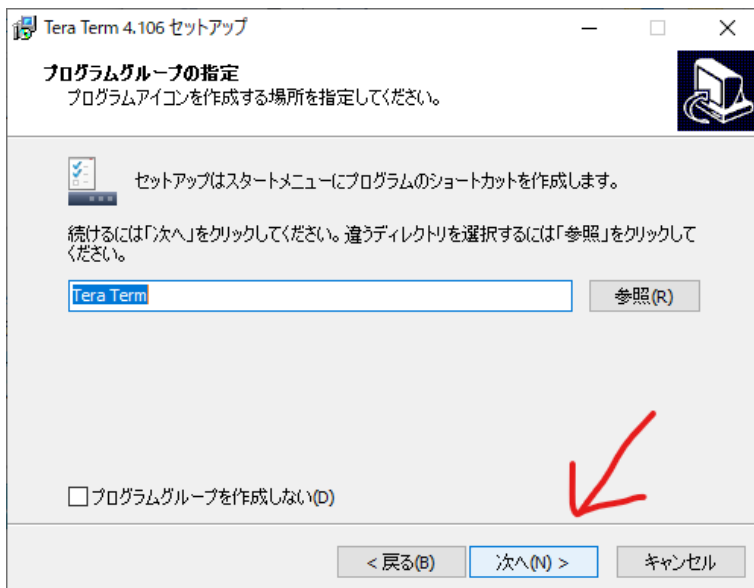
次のコンポーネントの選択では、初期値のままでいく。



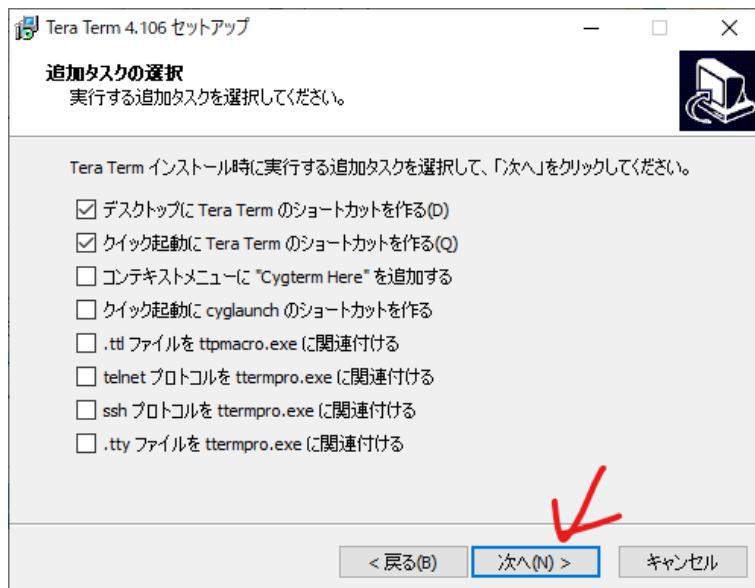
言語の選択では、“日本語”を選択。



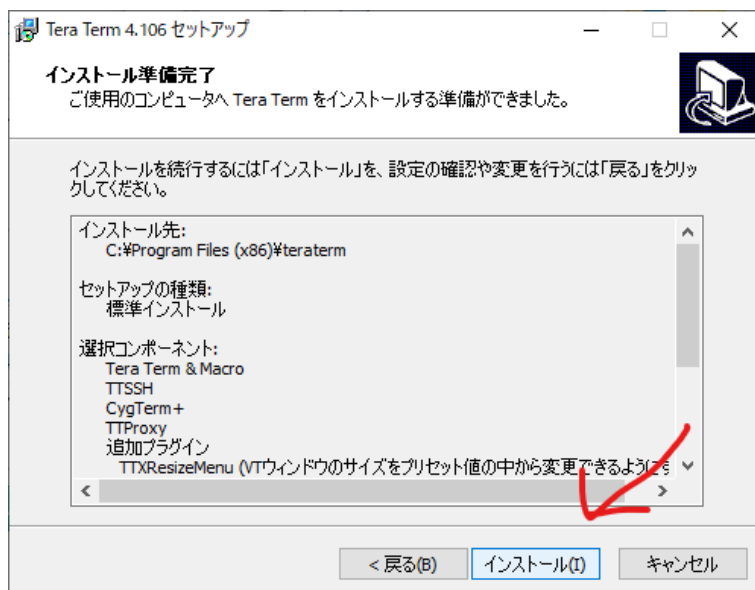
プログラムグループの指定は、そのままです。



追加タスクの選択も、そのままをクリック。



確認画面。



完了。



3 Web サーバーを動かす

3.1 作業フォルダで HTML ファイルを作成

適当なフォルダに作業フォルダを作る。

ここでは作業フォルダを test とする。

その test の中に、index.html を作成する。内容は以下。

リスト 1 index.html

```
1 <!doctype html>
2 <html lang="ja">
3   <head>
4     <meta charset="utf-8">
5     <title>TEST</title>
6   </head>
7   <body>
8     <h1>TEST</h1>
9   </body>
10 </html>
```

3.2 簡易 Web サーバーを起動

test フォルダでコマンドプロンプトを起動する。

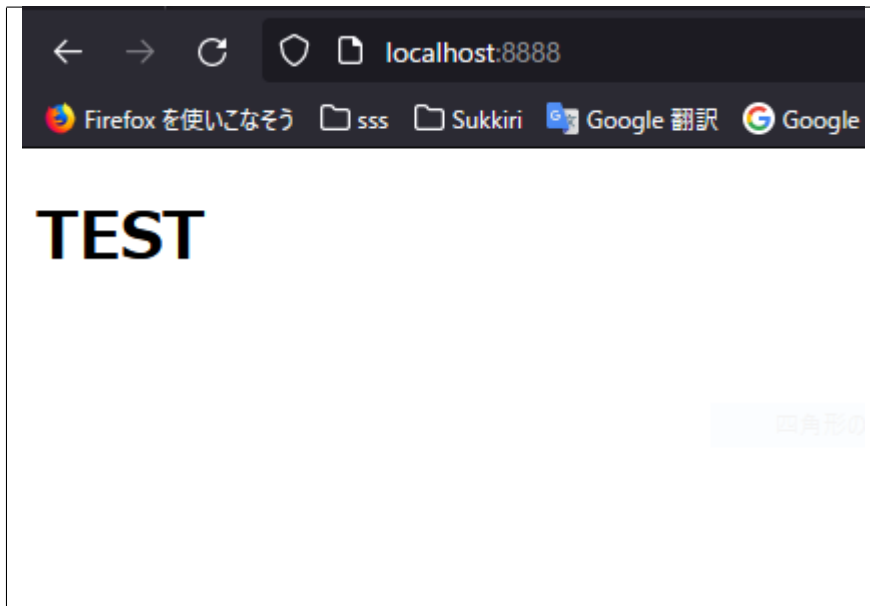
以下のコマンドを実行する。

```
> php -S localhost:8888
```

すると、以下のような文字列が表示され、Web サーバーが起動したことがわかる。

```
[Sat Aug ...(略)...] PHP 8.0.9 Development Server (http://localhost:8888) started
```

ブラウザで <http://localhost:8888/> にアクセスしてみる。



先ほど作成した index.html が表示される。

4 Telnet で Web サーバーと通信する

4.1 Telnet とは

Telnet とは、“ネットワークを経由して別のコンピュータに接続し、そのコンピュータを操作する”ためのプロトコルである。

そのプロトコルを使った通信のためのツールが telnet (同じ名前)。

昔からあるツールだけど、現在でも、いろいろなサーバを管理するのに、そのサーバに接続するためのツールとして使われている。

ここでいう“接続”とは、ネットワーク越しに離れたコンピュータにコマンドを送ったりすること。つまり、“通信する”ということ。

“telnet.exe”というプログラムが Windows にはあるけれど、キョーレツ使いにくいので、TeraTerm を使う。

4.2 TeraTerm の設定

まず、使うための準備が必要である。

TeraTerm がインストールされたフォルダ C:\Program Files (x86)\teraterm を開ける。

その中に、TERATERM.INI というファイルがあるので、念のため、バックアップのためのコピーを作っておいて (“TERATERM.INI.org” というファイル名にしておく)、TERATERM.INI を TeraPad で開く。

```
TCPLocalEcho=off
TCPCRSend=
```

と書かれた箇所を検索して探し出し、(たぶん 680 行目くらい)、以下のように修正する。

```
TCPLocalEcho=on
TCPCRSend=CRLF
```

以下のように修正する。

```
673 ; >standard telnet port↓
674 TelPort=23↓
675 ↓
676 ; Keep-Alive packet sending interval on telr
677 TelKeepAliveInterval=300↓
678 ↓
679 ; Auto setup for non-telnet↓
680 TCPLocalEcho=off↓
681 TCPCRSend=↓
682 ↓
683 ; Terminal Speed (telnet/SSH)↓
684 TerminalSpeed=38400↓
685 ↓
686 ; Terminal Unique ID↓
687 TerminalUID=FFFFFFFF↓
688 ↓
689 ; Title format↓
```



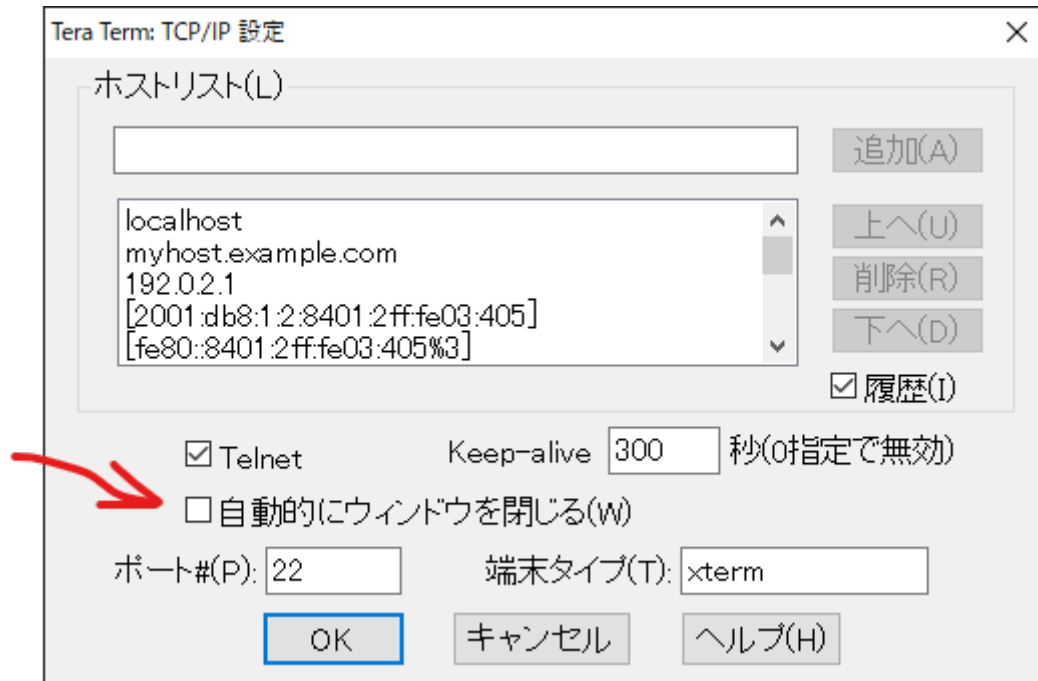
```
673 ↓
676 ; Keep-Alive packet sending interval
677 TelKeepAliveInterval=300↓
678 ↓
679 ; Auto setup for non-telnet↓
680 TCPLocalEcho=on↓
681 TCPCRSend=CRLF↓
682 ↓
683 ; Terminal Speed (telnet/SSH)↓
684 TerminalSpeed=38400↓
685 ↓
686 ; Terminal Unique ID↓
687 TerminalUID=FFFFFFFF↓
688 ↓
```

次に TeraTerm を起動する。

起動したら、“キャンセル”として、以下の設定をおこなう。

“設定” —— “TCP/IP” と選択する。

“自動的にウィンドウを閉じる” のチェックをはずす。



“設定” —— ” 設定の保存” を選択。 “TERATERM.INI” に上書き保存する。

TeraTerm を終了する。

5 Telnet で GET リクエスト

5.1 ブラウザは何をやっているのか？

現在、test フォルダを公開フォルダとして Web サーバーが動作している。

ブラウザから `http://localhost:8888` とすると、このフォルダにある `index.html` を表示させることができる。

このとき、ブラウザは Web サーバーにどういう働きかけをしているのか？

また、Web サーバーはどのような返答をしているのか？

ブラウザのやっていることを Telnet でやってみることができる。

5.2 GET リクエスト

ブラウザから、あるサイトにアクセスして、そのページを表示させたいとする。

これをあるがままに言うと、ブラウザがあるサイトにアクセスして、

このページの HTML をダウンロードさせてくれ

という要求をおこなっていることになる。

これを GET リクエスト という。

GET リクエスト のやり方は決っていて、以下のようなコマンドを送る。

```
GET / HTTP/1.1
Host: localhost:8888
(空行)
```

1 行目は、GET リクエストであることを示す。“/” は、パスをあらわす。“/index.html” を指定したと同じ。“HTTP.1.1” は、HTTP の 1.1 バージョンで通信するという意味。

2 行目は、相手ホスト名を指定している。80 番ポート (普通はこのポート) なら省略できるが、我々はさきほど、この Web サーバーを 8888 番ポートで起動したから、このポートを指定している。

そして、3 行目は、空行 を送る。これは絶対必要。

最後に <Enter> を送る。

この空行までをヘッダ部という。これが GET リクエストの決まりである。

これを TeraTerm でやってみる。

5.3 TeraTerm で Get リクエスト

TeraTerm を起動する。

“ホスト” に localhost と指定する。

“サービス” は Telnet にチェックを入れる。

“TCP ポート” は 8888 と指定する。

ほかはそのままで、“OK” とする。

Tera Term: 新しい接続

☒ TCP/IP ホスト(T): localhost

☒ ヒストリ(O)

サービス: ☒ Telnet TCPポート#(P): 8888

☐ SSH SSHバージョン(V): SSH2

☐ その他 IPバージョン(N): AUTO

☐ シリアル(E) ポート(R):

黒い画面があらわれるので、以下の内容を黒い画面に入力する。

```
GET / HTTP/1.1
Host: localhost:8888
(空行)
```

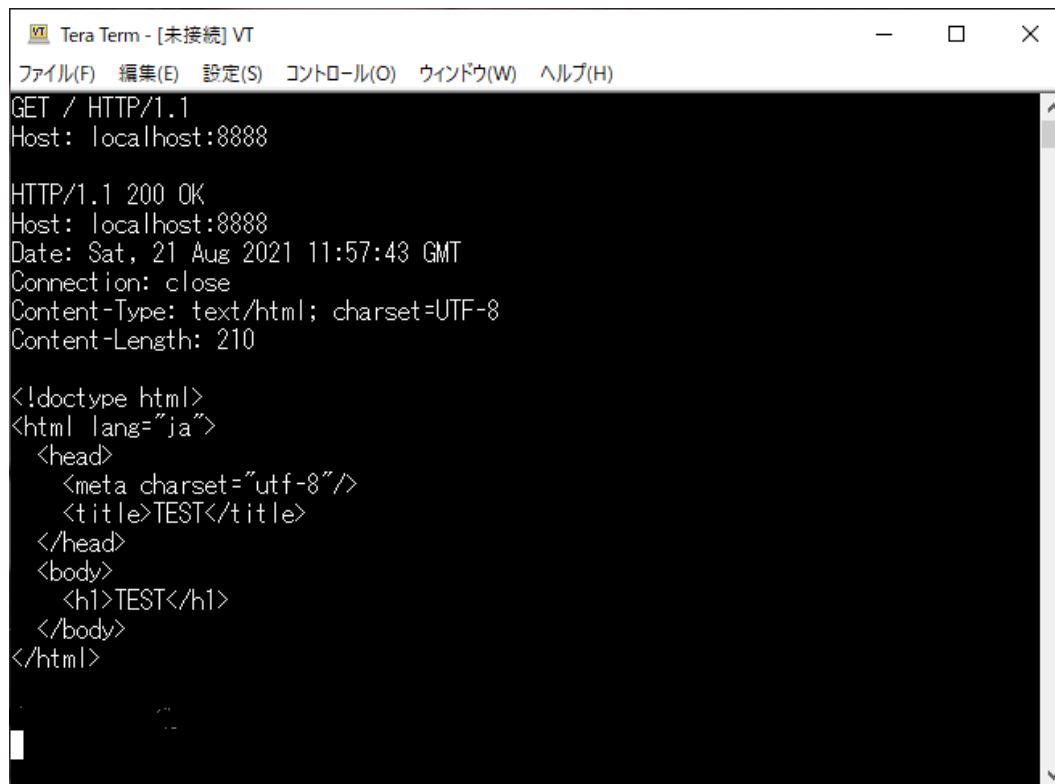
大文字、小文字に気をつけて入力する。

localhost - Tera Term VT

ファイル(F) 編集(E) 設定(S) コントロール(O) ウィンドウ(W) ヘルプ(H)

```
GET / HTTP/1.1
Host: localhost:8888
```


そして <Enter> キーを押すと、以下のように出力される。



```
Tera Term - [未接続] VT
ファイル(F) 編集(E) 設定(S) コントロール(O) ウィンドウ(W) ヘルプ(H)
GET / HTTP/1.1
Host: localhost:8888

HTTP/1.1 200 OK
Host: localhost:8888
Date: Sat, 21 Aug 2021 11:57:43 GMT
Connection: close
Content-Type: text/html; charset=UTF-8
Content-Length: 210

<!doctype html>
<html lang="ja">
  <head>
    <meta charset="utf-8"/>
    <title>TEST</title>
  </head>
  <body>
    <h1>TEST</h1>
  </body>
</html>
```

空行の下の “HTTP...” 以下は Web サーバーからの応答 (レスポンス) である。

- HTTP/1.1 — HTTP プロトコルのバージョン 1.1 で応答するということ。
- 200 OK — 正常な処理だということ。
- Host: localhost:8888 — サーバー側のホスト名とポート番号。
- 日付 – ”GMT” となっているのは、グリニッジ標準時のこと。日本よりも 9 時間遅い。
- Connection: close — この応答で接続が閉じられたことを表す。
- Content-Type: text/html; charset=UTF-8 — 空行の後に送る文字列 (ボディ部) は HTML で、文字コードは UTF-8 であることを (ブラウザに) 伝えている。
- Content-Length: 210 — 空行の後に送る文字列は 210 文字であることを示している。
210 文字というのは、本当は 159 文字で、この画像では消しているが、</html> の後に注釈の文字列が続いていたのである。

ここまでの ” ヘッダ部 ” で、 ” 空行 ” に区切られて、以下 ” ボディ部 ” が続く。

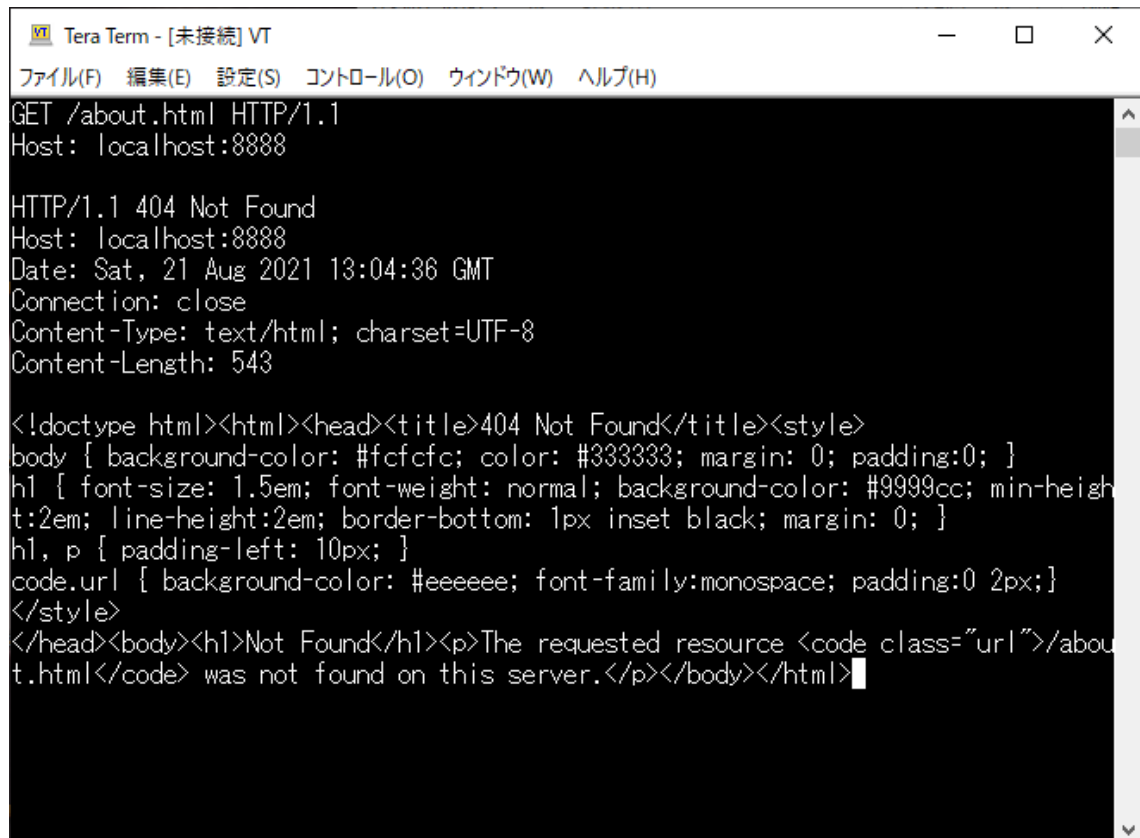
ボディ部はブラウザが受け取って処理をする。つまり、文字や画像をブラウザの画面に表示するのである。それを レンダリング (描画) という。

5.4 さまざまな GET リクエスト

Web サーバーに送る GET リクエストでは、“/” のみを指定したが、それ以外にたとえば、“/menu.html” などとファイル名 (パス) を指定することが多い。

もし、存在しないファイル名を指定すると、どうなるか？

これは、存在しないファイル “about.html” を指定した例である。



```
VT Tera Term - [未接続] VT
ファイル(F) 編集(E) 設定(S) コントロール(O) ウィンドウ(W) ヘルプ(H)
GET /about.html HTTP/1.1
Host: localhost:8888

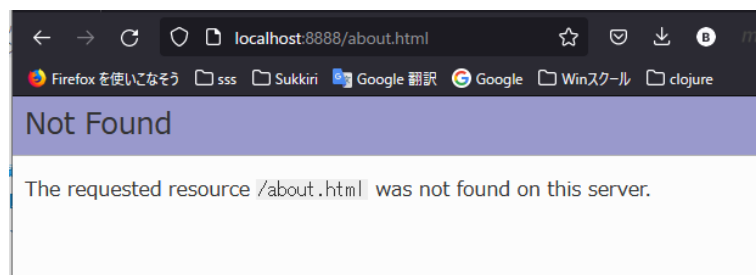
HTTP/1.1 404 Not Found
Host: localhost:8888
Date: Sat, 21 Aug 2021 13:04:36 GMT
Connection: close
Content-Type: text/html; charset=UTF-8
Content-Length: 543

<!doctype html><html><head><title>404 Not Found</title><style>
body { background-color: #fcfcfc; color: #333333; margin: 0; padding: 0; }
h1 { font-size: 1.5em; font-weight: normal; background-color: #9999cc; min-height: 2em; line-height: 2em; border-bottom: 1px inset black; margin: 0; }
h1, p { padding-left: 10px; }
code.url { background-color: #eeeeee; font-family: monospace; padding: 0 2px; }
</style>
</head><body><h1>Not Found</h1><p>The requested resource <code class="url">/about.html</code> was not found on this server.</p></body></html>
```

サーバーからの応答の 1 行目に、

HTTP/1.1 404 Not Found

とある。“404 エラー” と呼ばれる応答例である。これをブラウザで表示すると、以下のようになる。



5.5 ブラウザのやっていること

ブラウザは、ユーザーが URL 欄にアドレスを入力したら、そのアドレスに対して GET リクエストを送信する。

Web サーバーは、クライアントであるブラウザから URL を受け取って、そのファイルが存在するなら “200” という番号をヘッダ部書き、ファイル本体をボディ部に書いて、ブラウザに送る。

もしそのファイルが存在しなかったら、“404” という番号をヘッダ部書き、“Not Found” と表示させるための文字列をボディ部に書いて、ブラウザに送る。

Web サーバーから送られてきた応答のうち、ヘッダ部は、Web サーバーとブラウザとの暗黙のやりとりであり、ブラウザには直接表示されない。確認するためには、ブラウザのデベロッパーツールの” ネットワーク” タブを見るとよい。ボディ部はブラウザが画面にレンダリングするための情報である。デベロッパーツールの” インスペクター” タブで確認できる。

ボディ部は HTML という約束事にしたがって書かれている。(HTML のなかにも <head> 部と <body> 部があるが、混乱しないように。)

6 Telnet で POST リクエスト

6.1 サーバー側で準備をする

今度は POST 送信をやってみる。

まず、サーバー側に以下のような HTML を用意する。

リスト 2 post.html

```
1 <!doctype html>
2 <html lang="ja">
3   <head>
4     <meta charset="utf-8"/>
5     <title>Post</title>
6   </head>
7   <body>
8     <h1>Post</h1>
9     <form action="receive.php" method="post">
10       <label for="name">お名前</label>
11       <input type="name" name="name" id="name" />
12       <input type="submit" value="送信"/>
13     </form>
14   </body>
15 </html>
```

クライアント (ブラウザ) から、以下のように GET リクエストが送られると.....、

```
GET /post.html HTTP/1.1
Host: localhost:8888
(空行)
```

以下のようにレスポンスが返ってくる。

```
Tera Term - [未接続] VT
ファイル(F) 編集(E) 設定(S) コントロール(O) ウィンドウ(W) ヘルプ(H)
GET /post.html HTTP/1.1
Host: localhost:8888

HTTP/1.1 200 OK
Host: localhost:8888
Date: Sat, 21 Aug 2021 22:15:09 GMT
Connection: close
Content-Type: text/html; charset=UTF-8
Content-Length: 358

<!doctype html>
<html lang="ja">
  <head>
    <meta charset="utf-8"/>
    <title>Post</title>
  </head>
  <body>
    <h1>Post</h1>
    <form action="receive.php" method="post">
      <label for="name">お名前</label>
      <input type="name" name="name" id="name" />
      <input type="submit" value="送信"/>
    </form>
  </body>
</html>
```

```
HTTP/1.1 200 OK
Host: localhost:8888
Date: Sat, 21 Aug 2021 22:15:09 GMT
Connection: close
Content-Type: text/html; charset=UTF-8
Content-Length: 358

<!doctype html>
<html lang="ja">
  <head>
    <meta charset="utf-8">
    <title>Post</title>
  </head>
  <body>
    <h1>Post</h1>
    <form action="receive.php" method="post">
      <label for="name">お名前</label>
      <input type="text" name="name" id="name" >
      <input type="submit" value="送信">
    </form>
  </body>
</html>
```

ブラウザの画面には以下のようにレンダリングされる。



ここでユーザーは名前を入力し、送信ボタンを押して、Web サーバーに入力した情報を送信する。
このとき、post.html には、以下のように送り先が指定されている。

```
<form action="receive.php" method="post">
```

これは、送り先が “receive.php” で、送信方法が “post” 送信であるという指定である。

まず、Web サーバーに、クライアント (ブラウザ) から送信された情報を受け取るためのファイル “receive.php” を作成する。“test” フォルダに以下の内容で作成する。

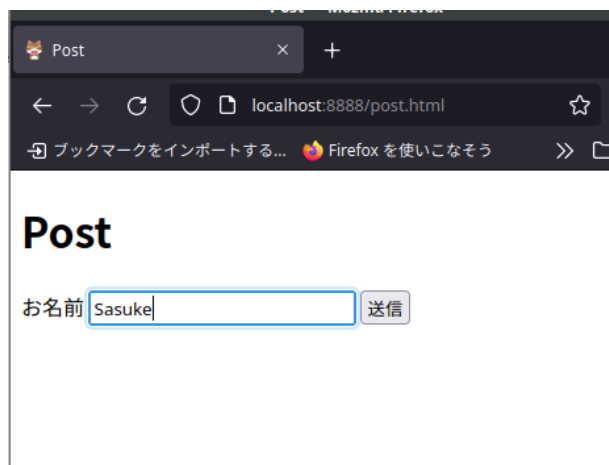
リスト 3 receive.php

```
1 <?php
2 $name = $_POST['name'];
3 ?>
4 <!doctype html>
5 <html lang="ja">
6   <head>
7     <meta charset="utf-8"/>
8     <title>receive</title>
9   </head>
10  <body>
11    <h1>receive</h1>
12    <p>name="<?php echo $name; ?>"</p>
13  </body>
14 </html>
```

これは、クライアントから送られてきた “name” の内容を \$name という変数にセットし、それをブラウザで表示するというものである。

実際に試してみる。“test”フォルダでコマンドプロンプトを起動し、`> php -S localhost:8888`として、Web サーバーを起動する。

ブラウザで `http://localhost:8888/post.html` にアクセスして、名前を入力して送信する。



名前を入力して送信すると.....



このようにブラウザの画面で表示される。

これを TeraTerm を使って Telnet 画面でやってみる。そのことで、ブラウザと Web サーバーでは実際にどのようなやりとりが行われているかを確認する。

6.2 Telnet での POST 送信のしかた

“test”フォルダで Web サーバーが動作している状態で、TeraTerm を起動し、POST 送信をおこなうが、今度は送信文字列が長いので、TeraPad などのエディタで、送信文字列をあらかじめ作成して、それをコピーして TeraTerm の画面に貼り付けることにする。

リスト 4 post-request.txt

```
1 POST /receive.php HTTP/1.1
2 Host: localhost:8888
3 Content-Length: 11
4
5 name=Sasuke
```

1. 今回は POST 送信なので “POST” と指定する。
また、送り先として “receive.php” を指定する。
2. これは GET の場合と同じである。
3. ここでは、ボディ部の長さを伝えている。“name=Sasuke” で 11 バイトである。
4. 空行がヘッダ部とボディ部の区切りである。
5. “name” という変数 (コントロール名) に “Sasuke” という値 (文字列) をセットしている。

5 行目の “name=Sasuke” は、post.html の以下の部分の記述に相当する。

```
<input type="text" name="name" id="name" >
```

つまり、ユーザーが入力してくれた 値 (文字列) に “name” という名前をつけて Web サーバーに送るという意味である。

ここでは、“name” という文字列がたくさん出てきてややこしいが、たとえば、以下のようになっていると

.....

```
<input type="text" name="namae" id="name" >
```

post-request.txt の 5 行目は、以下ようになる。

```
namae=Sasuke
```

要するに、Web サーバーから送られてきた post.html を画面に表示し、ユーザーが名前を入力し、送信ボタンが押されたとき、

```
POST /receive.php HTTP/1.1
Host: localhost:8888
Content-Length: 11

name=Sasuke
```

このようなコマンド文字列が Web サーバーに送られているのである。

そして、POST 送信の特徴は、送信されるデータが ボディ部 に埋め込まれるということである。ということとは、SSL 通信では暗号化されるということになる。(ヘッダ部は暗号化されない)

今回は送るデータは一つだけだが、POST 送信では多くのデータを送ることができる。

POST 送信以外に、GET 送信でもデータを送ることができる。以下のようなやりかたで送信する。


```
GET /receive.php?name=Sasuke HTTP/1.1
```

今回は GET 送信でのデータ送信は詳しくは触れない。

6.3 TeraTerm で POST 送信をやってみる

それでは、TeraTerm で POST 送信をやってみる。今回は、実際のやりとりを再現するために、“ローカルエコー”を“OFF”にしてみる。

今までは、TeraTerm の画面では、我々の入力した文字列と、サーバーが送り返してきた文字列を両方とも表示していた。しかし実際は、我々が入力した文字列はサーバーに送られてしまい、我々は見ることができないはずである。それでは仕事がやりにくいので、telnet プログラムには、ユーザーが入力した文字列をサーバーに送ると同時に、ユーザーにも表示してくれる機能がついている。それが“ローカルエコー”で、それを“ON”にすることで実現できる。

今までは TeraTerm の画面に直接文字列を入力していたので、“ローカルエコー”が“ON”のほうがやりやすかった。今回は、エディタで文字列を準備しておいて、TeraTerm の画面には貼り付けるだけなので、“ローカルエコー”を“OFF”にしてもできる。

“ローカルエコー”を“OFF”にするには、2つの方法がある。

1. 設定ファイル (TERATERM.INI) を書き変える。
2. 一時的に変更する。

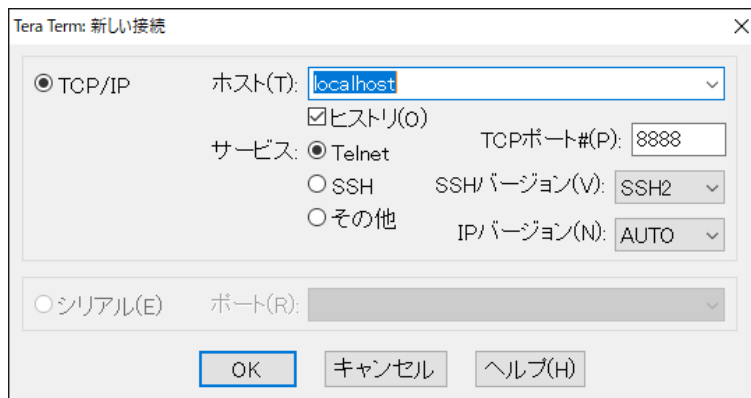
今回は、“一時的に変更する”のほうでやってみる。

手順をまとめると以下になる。

1. TeraTerm を起動する。
2. “設定” — “端末”を選択して“端末の設定”画面で、“ローカルエコー”のチェックをはずす。
3. “post-request.txt”の内容をコピーする。
4. “編集” — “貼り付け”を選択。
5. 貼り付ける文字列の確認画面が開くので、それを OK する。
6. 黒い画面になるので、“<Enter>”キーを押す。
7. サーバーから文字列が送られてくる。

以下、ひとつひとつの手順を確認する。

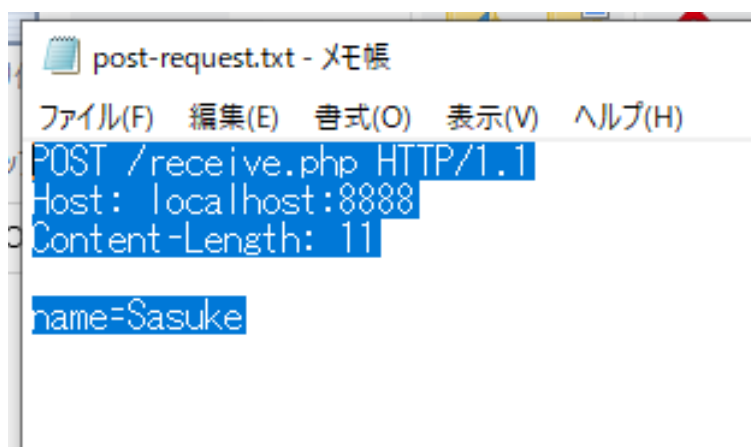
1. TeraTerm を起動する。



2. “設定” — “端末” を選択して “端末の設定” 画面で、“ローカリエコー” のチェックをはずす。

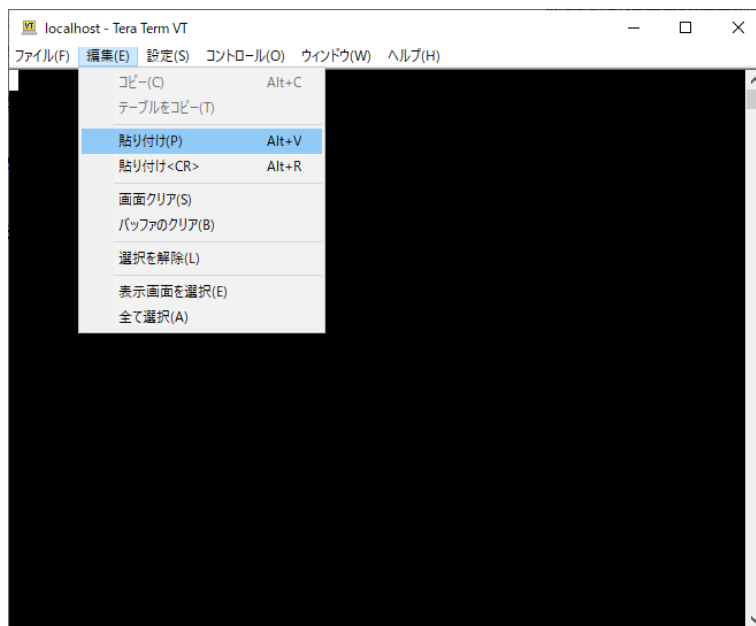


3. “post-request.txt” の内容をコピーする。

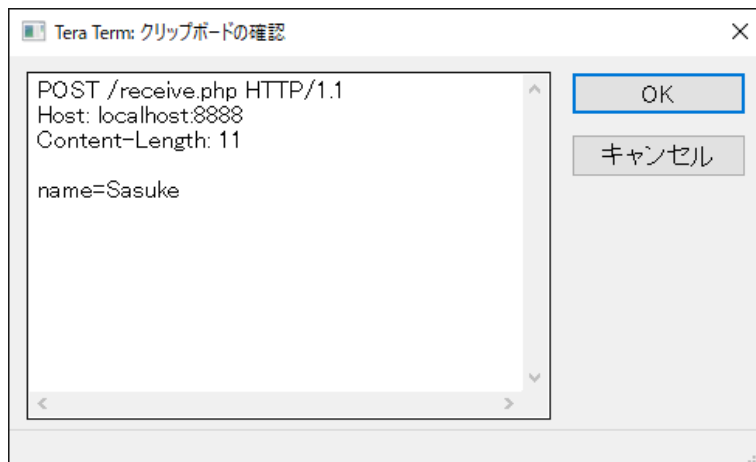


余分な”空行”などコピーしないように “Sasuke” までをコピーする。

4. TeraTerm の黒い画面で “編集” — “貼り付け”

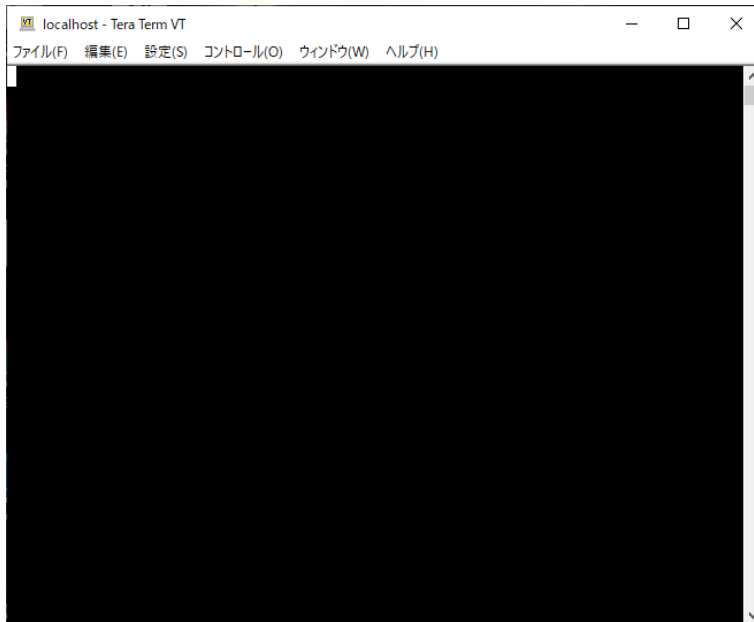


5. 貼り付ける文字列の確認画面が開くので、それを OK する。

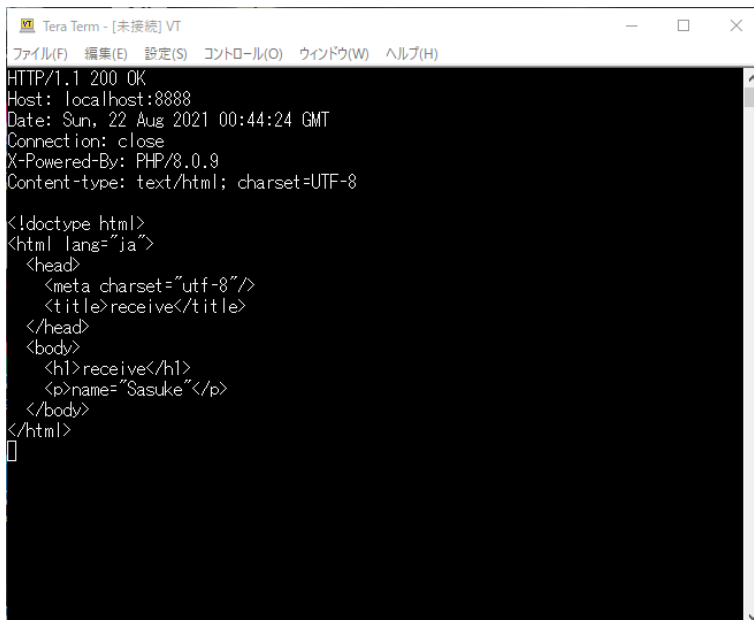


6. 黒い画面になるので、“<Enter>” キーを押す。

こちらから貼り付けた文字列はサーバーへ送られたので、画面には出ていない。これが“ローカルエコー・オフ”ということである。



7. サーバーから文字列が送られてくる。



送られてきた文字列は、以下である。

```
HTTP/1.1 200 OK
Host: localhost:8888
Date: Sun, 22 Aug 2021 03:15:13 GMT
Connection: close
X-Powered-By: PHP/8.0.9
Content-type: text/html; charset=UTF-8

<!doctype html>
<html lang="ja">
  <head>
    <meta charset="utf-8"/>
    <title>receive</title>
  </head>
  <body>
    <h1>receive</h1>
    <p>name="Sasuke"</p>
  </body>
</html>
```

ヘッダ部は GET 送信のときとそんなに変りはない。5 行目で PHP で処理したことが明記されている。ボディ部のほうは、“receive.php” の内容であるが、`<?php ... ?>` で記述された部分はない。特に、`<?php echo $name; ?>` で記述された部分は 文字列”Sasuke” に置き変わっている。

7 PHP と JavaScript

7.1 PHP のはたらき

PHP は、ユーザーがブラウザを通じて入力したデータを扱うことができる。また、データベースに接続して、そこから取得したデータを扱うことができる。PHP は、Web サーバーと連携して動作し、データを HTML の中に埋め込むことができる。

だから PHP は、サーバーで動く。データを加工し、HTML に埋め込む。サーバーからクライアント (ブラウザ) に応答するときには、PHP はもう姿を消している。

7.2 JavaScript のはたらき

では、JavaScript はどうだろうか？

ために、以下のような HTML を書いてみる。そこに JavaScript を埋め込んでみる。

リスト 5 sample.html

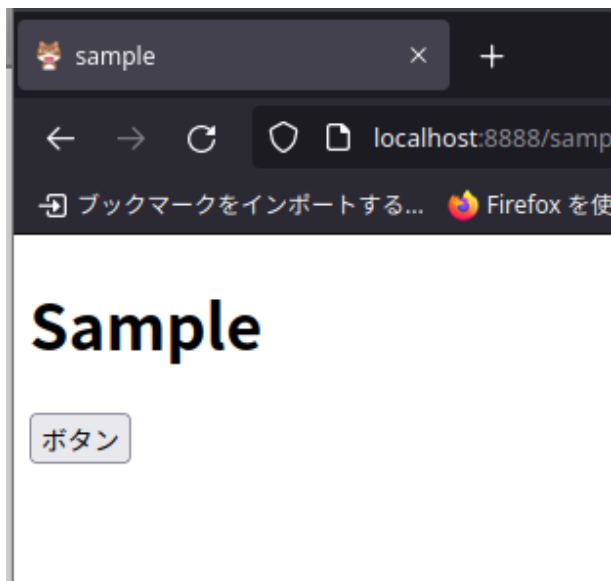
```
1 <!doctype html>
2 <html lang="ja">
3   <head>
4     <meta charset="utf-8">
5     <title>sample</title>
6   </head>
7   <body>
8     <h1>Sample</h1>
9     <button id="button">ボタン</button>
10    <script>
11      const btn = document.getElementById('button')
12      btn.onclick = function () {
13        alert('Hello!')
14      }
15    </script>
16  </body>
17 </html>
```

上のコードを “test” フォルダに “sample.html” という名前で保存する。

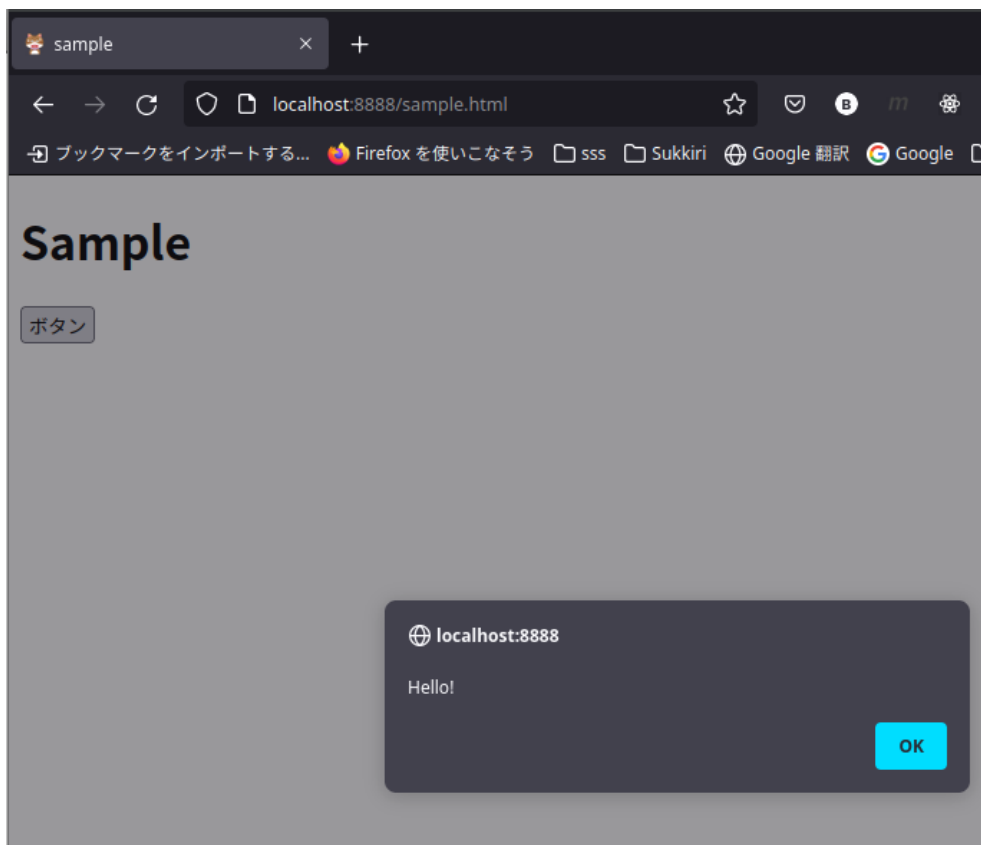
そして、“test” フォルダで Web サーバーを起動する。(起動しっぱなしになっていてもよい)

Web サーバーの起動は、 `> php -S localhost:8888` で起動できる。

ブラウザで “http://localhost:8888/sample.html” にアクセスする。



このボタンをクリックすると、



と表示される。

7.3 TeraTerm でやってみる

TeraTerm で GET してみたら、どうだろうか？

Web サーバーが動作しているのを確認したのち、TeraTerm を起動する。

ホスト : localhost

サービス : Telnet

TCP ポート : 8888

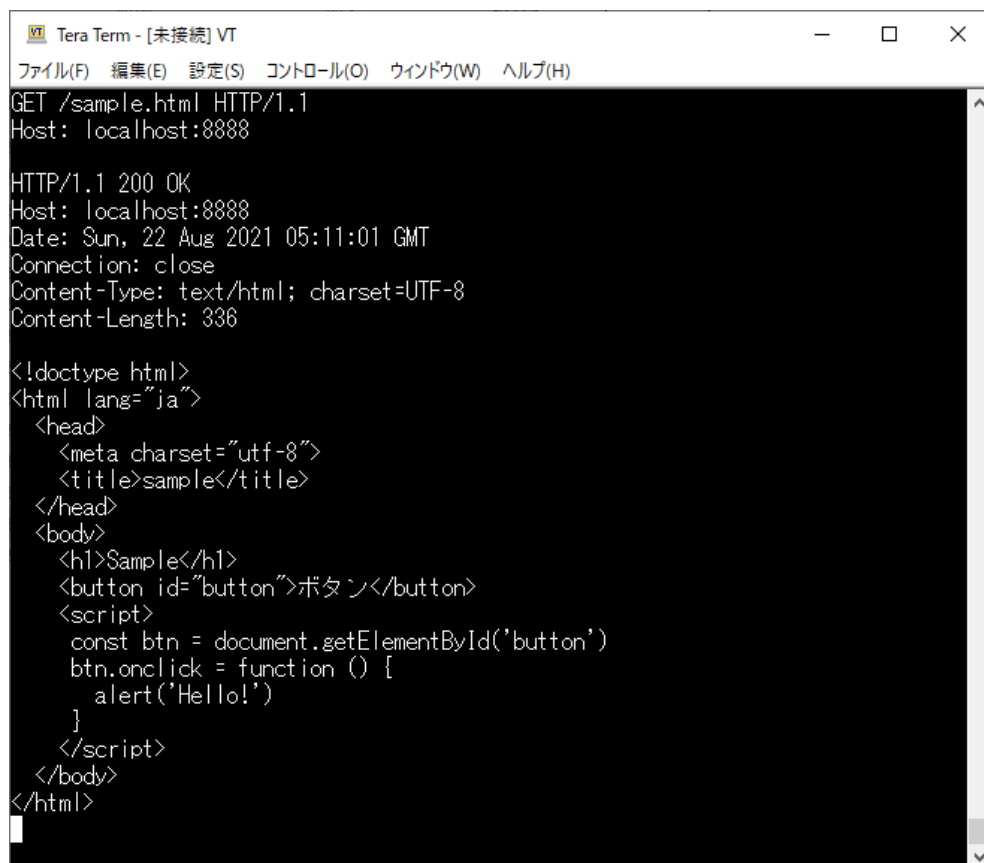
開いた黒い画面に以下を入力する。

```
GET /sample.html HTTP/1.1
```

```
Host: localhsot:8888
```

```
(空行)
```

ここで <Enter> とすると、以下のように出力される。



```
Tera Term - [未接続] VT
ファイル(F) 編集(E) 設定(S) コントロール(O) ウィンドウ(W) ヘルプ(H)
GET /sample.html HTTP/1.1
Host: localhost:8888

HTTP/1.1 200 OK
Host: localhost:8888
Date: Sun, 22 Aug 2021 05:11:01 GMT
Connection: close
Content-Type: text/html; charset=UTF-8
Content-Length: 336

<!doctype html>
<html lang="ja">
  <head>
    <meta charset="utf-8">
    <title>sample</title>
  </head>
  <body>
    <h1>Sample</h1>
    <button id="button">ボタン</button>
    <script>
      const btn = document.getElementById('button')
      btn.onclick = function () {
        alert('Hello!')
      }
    </script>
  </body>
</html>
```



```
HTTP/1.1 200 OK
Host: localhost:8888
Date: Sun, 22 Aug 2021 05:07:08 GMT
Connection: close
Content-Type: text/html; charset=UTF-8
Content-Length: 336

<!doctype html>
<html lang="ja">
  <head>
    <meta charset="utf-8">
    <title>sample</title>
  </head>
  <body>
    <h1>Sample</h1>
    <button id="button">ボタン</button>
    <script>
      const btn = document.getElementById('button')
      btn.onclick = function () {
        alert('Hello!')
      }
    </script>
  </body>
</html>
```

このように JavaScript は HTML といっしょに Web サーバーから送られてくる。そして、ユーザーのブラウザでレンダリングされ、そのときに JavaScript は動作を始める。

8 参考

telnet で Web サーバに接続してみよう:TCP/IP アレルギー撲滅ドリル