PHP からブラウザへ最も基本的なテキストを渡す事を指示したい場合、**Content-Type: text/plain** と記述されます。これを受け取るとブラウザは単純なテキストとしてデータを表示しようとします。

以下に代表的なデータの種類を列挙します

1) [　　] ： ブラウザにとって最も一般的な HTML
2) [　　] ： 特に種類を指定しないバイナリデータ(ダウンロードダイアログが開く)
3) [　　] ： 自然画像フォーマットである、拡張子が通常 .jpg のファイル

また、テキストファイルである場合、そのテキストを表現する為のキャラクタセットがSHIFT_JISである場合は Content-Type に [　　] =Shift_JISを追加します

現在のWEB では、キャラクタセットは **UTF-8** が標準になっています。しかし、 [　　] **の基本キャラクタセット**は Shift_JIS なので、WEBアプリケーション開発ではキャラクタセットの存在を意識する必要があります。

あらゆる開発言語で、処理を共通化したり機能別に独立させる事が行われます。クラスを作成する場合、メソッドと呼ばれる実行単位がそれにあたりますが、PHP ではクラスを作成可能ではありますが、一般的には [　　] と言う**キーワード**で、ユーザー定義関数を作成します。この記述方法は、**JavaScript** と同様で、関数名と引数を定義して使用するものです。

しかし、PHP の関数では他の言語と全く違う扱いをするのが [　　] 変数の扱いです。他の言語では、関数の外側で定義または発生した変数は、関数内で参照可能です。しかし、PHP では [　　] キーワードを使用してその変数名を宣言しないと参照できない事に注意して下さい(**スーパーグローバル変数**は常に参照可能です)

WEBアプリケーションを作成する上で、**プログラマが昔から意識し続けなければならなかった**のが、ブラウザの [　　] の問題です。ブラウザは利用するユーザに対してより良好なレスポンスを提供する必要があったので、ブラウザに表示されるデータは全てが最新では無く PC に保存されているものもあります。しかし、WEBアプリケーションでは動的に情報を作成する為、ユーザに表示するデータは常に最新である必要があります。

このような対処には、ブラウザに指示を出す関数でコントロールしますが、PHP のある処理を組み込む事によって比較的容易にその対応を行う事ができます。それが [　　] の処理です

```
session_cache_limiter('nocache');
session_start();
```

session_start(); は、本来 [　　] という**スーパーグローバル変数**を利用する為の開始処理ですが、そのオプションとして session_cache_limiter('nocache'); を実行しておくと、ブラウザに対して PC に保存してはならないという指示が与えられます。この時利用可能になるスーパーグローバル変数は、通常 [　　] 中のデータ保持に使用されます

PHP はサーバで動作しますが、ユーザが利用するのはブラウザであり、自分のすぐ近くにある PC の CPU で処理されたものです。よって WEBアプリケーションを作成するには**画面のレイアウト**を定義する為の [　　] で、サーバへデータへ送る為の入力画面を作成する必要があります。その際、サーバの為に必須となる要素が **FORM** です。そして、その中に記述されたものが送信データとなります。但し、送信データとなる要素は限られており、[　　] 要素が最も良く使用され、**type 属性**によってその利用方法も変化します。

1) **text**　　改行のない通常のテキスト入力
2) [　　] **複数の選択肢より一つだけ**選択できるコントロール
3) [　　] 選択するかしないかの**２択**を表現するコントロール
4) [　　] 内容は text と同じだが**画面には表示されない**コントロール
5) [　　] いわゆる**カレンダーコントロール**
6) [　　] **ファイルアップロード**に使用するコントロール
7) [　　] **送信用ボタン**

７番目は、**サーバへデータを送信する為のボタン**となり、 [ ] 属性に指定した値が**ボタンの文字列**となります。そして、データをサーバへ送る為のコントロールに必ず必要な属性が [ ] 属性であり、**この属性が無ければサーバへは送られません**。また、この属性に指定した値がサーバ側の**スーパーグローバル変数のインデックス部分**にセットされる事になっています。

あと、画面よりサーバへデータを送る要素としてとても重要なコントロールがあり、一般的には『**コンボボックス**』と呼ばれますが、**要素名としては** [ ] となります。この要素は [ ] 属性を指定するといわゆる『**リストボックス**』というコントロールに変化します。そして、この要素の中に [ ] 要素を指定して**選択肢を設定**する事ができます。この選択肢には、 [ ] 属性で**サーバへ送りたい値を設定**しておき、利用するユーザにはテキストで意味を表現して**この要素の終了要素で挟んで**表示させます。

最後に、複数改行を入力可能ないわゆる**文章を入力可能**な要素が [ ] です。**初期値はこの要素の開始要素と終了要素の間に挟んで準備**する事が可能です。

PHP に対してデータを送る役割は **FORM 要素**ですが、この要素そのものが持つ属性が WEBアプリケーションにとって重要である事は言うまでもありません。

まず、 [ ] 属性によってサーバへのデータの送り方が大きく分けて２通りに分けられ、この属性を省略すると実行される通信方法を仮に『**アドレスバー方式**』と呼ぶことにします。この方式では、データは **URL の一部**として作成され、その URLそのものがサーバに到達します。この際、データ部分は [ ] と呼ばれ、データとデータの間は **&** で区切られます。

つまり、その URL は A 要素の [ ] 属性で指定可能なもので、データ部分が外部に容易に露出する事になるデータ通信方法です。

それに反してもう一つの通信方法は、ブラウザが直接サーバへデータを送る方法で、『**アドレスバー方式**』との大きな違いは送信可能なデータ量に対する制限が無い事です。この場合、FORM 要素の属性に **enctype** を使用して、値に [ ] を指定するとファイルのアップロードが可能になります。また、送信されたデータがアドレスバーに残る事も無く、確実にサーバへデータを送る事ができます。

そして、次に重要な属性は [ ] です。この属性を省略すると、データの送り先は現在表示している URL に対して送られます。つまり、省略せずに**他の PHP ファイルを指定すれば、その PHP にデータを送信可能**になります。こうする事によって、PHP で記述する処理を単純化する事が可能になり、プログラミングのコストを抑える可能性も大きくなります。

さらに、プログラムの処理そのものにはさほど影響は与えませんが、 [ ] 属性を使用する事によって、**ブラウザ上のどの部分に表示する**かを決める事ができます。例えば [ ] 要素をページ内に埋め込んで、PHP の送信結果をその中に表示する事が可能になります。また、この属性に値として [ ] を指定すると、送信のたびに新しいタブを開いてそのウインドウ内に結果が表示されるでしょう。

ここまでは、画面定義として決められた基本機能の指定方法ですが、WEB アプリケーションとして完成度を高くするには、このデータ送信機能を実行する前に**入力チェックを行う必要があります**。たとえ入力チェックが必要無い簡単な送信でも、ユーザの間違いを避ける為に確認ダイアログを表示して**送信のキャンセルをする事ができるようにする必要があります**。

その為の重要な FORM 要素の属性が [ ] です。このような属性は、他の要素でも同様で、頭に **on** が付き、JavaScript への橋渡しをするための属性です。よって、属性の値は JavaScript そのものが記述されます。

そして、FORM 要素のこの属性は送信のキャンセルを実現する為に最初に [ ] ステートメントを指定して、他で定義した関数の戻り値をシステムに返す事になっています。この他で定義した関数が返す値が [ ] の場合にサーバへの送信はキャンセルされます。