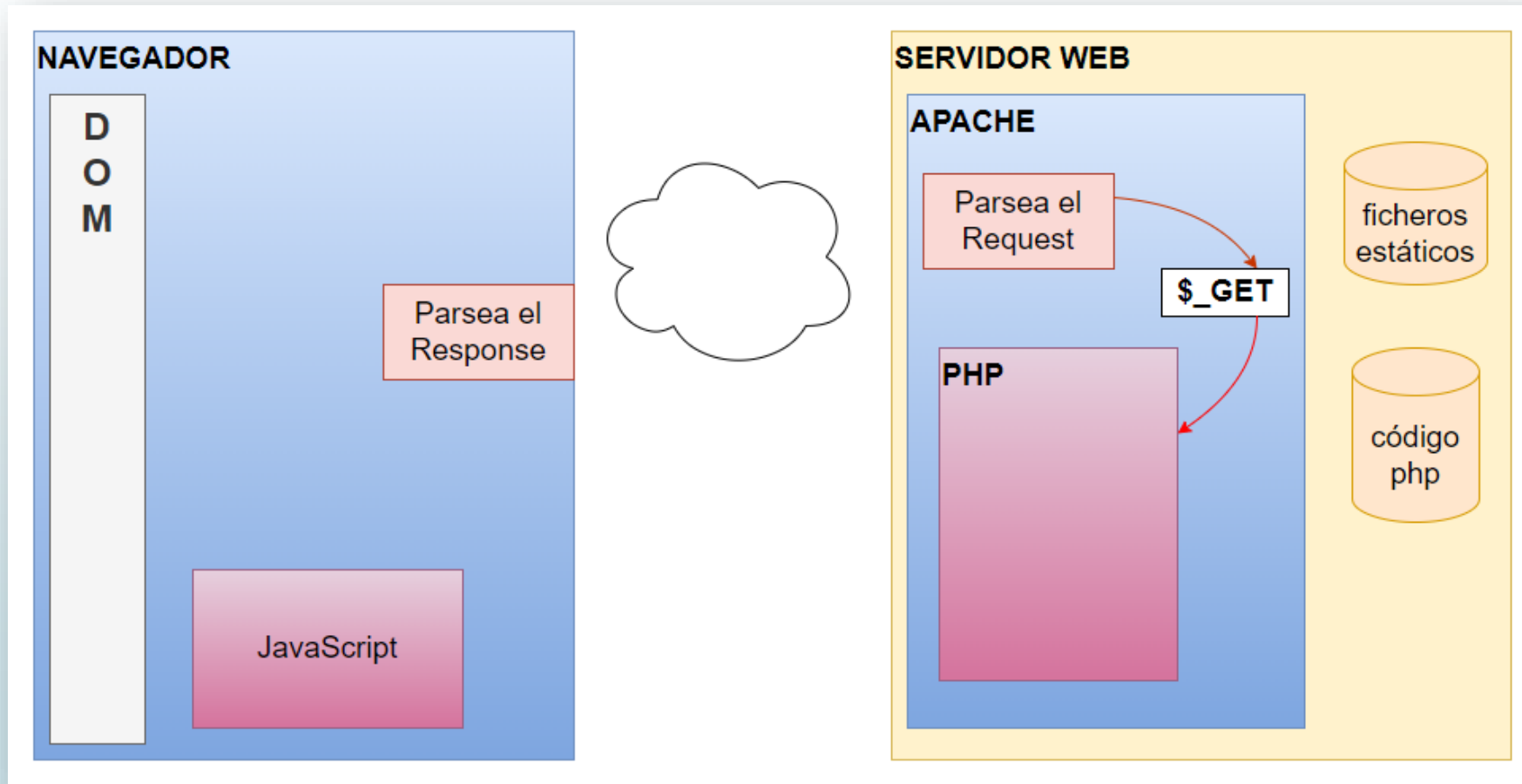




TEMA 3.2

Formularios en PHP

Interacción cliente-servidor



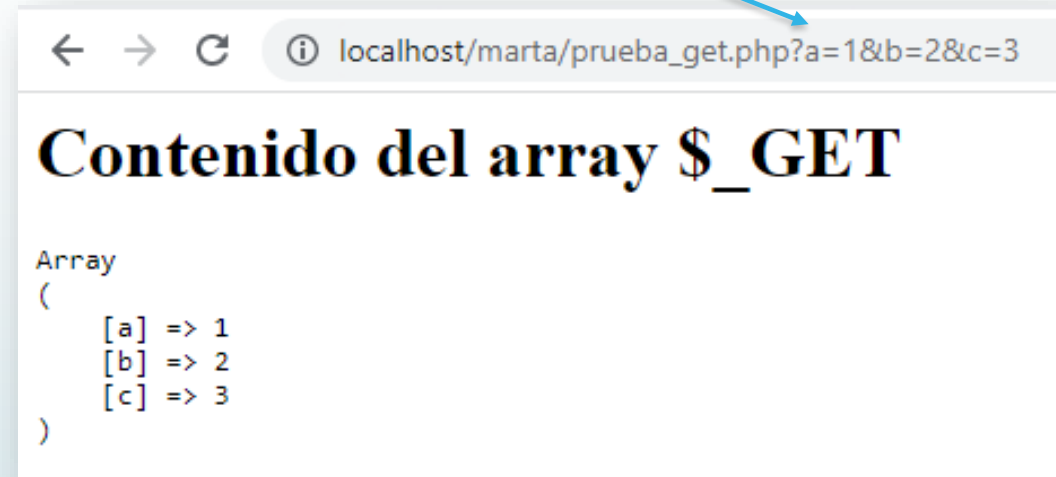
VARIABLES DE SERVIDOR

1. Como vimos en el tema 2, PHP almacena la información del servidor y de las peticiones HTTP recibidas en seis variables globales:
 1. **\$_ENV**: Variables de entorno del sistema
 2. **\$_GET**: Parámetros enviados con un submit con el método GET
 3. **\$_POST**: Parámetros enviados con un submit con el método POST
 4. **\$_COOKIE**: Contiene las cookies del request
 5. **\$_SERVER**: Contiene información sobre el servidor. Es un array asociativo, con las claves:
 1. PHP_SELF: Nombre del documento php que se está ejecutando
 2. SERVER_SOFTWARE: Por ejemplo, el Apache
 3. SERVER_NAME: El dominio, por ejemplo www.google.es
 4. REQUEST_METHOD: El método del request, por ejemplo GET
 5. REQUEST_URI: La URI sin el dominio
 6. QUERY_STRING: La lista de parámetros del request
 6. **\$_FILES**: Contiene información sobre los ficheros cargados con upload.

TRABAJANDO CON FORMULARIOS

- Parte del objetivo de PHP es **facilitar la interacción** con HTTP y HTTPS
- PHP **procesa la request** entrante en servidor y vuelca los datos en **variables superglobales**.
- Estas variables superglobales son normalmente arrays.
- Los parámetros vienen en las variables **\$_GET** y **\$_POST**, que son arrays asociativos
- **\$_GET** contiene los parámetros que llegan con la URL. La clave es el nombre del parámetro, el valor su valor
- En un request enviada con método GET, los parámetros se visualizan en la URL.
 - El símbolo "?" Indica el comienzo de los parámetros
 - El símbolo "=" asigna un valor al parámetro:
parametro=valor
 - El símbolo "&" indica que hay otro parámetro después.

Parámetros: **?a=1&b=2&c=3**

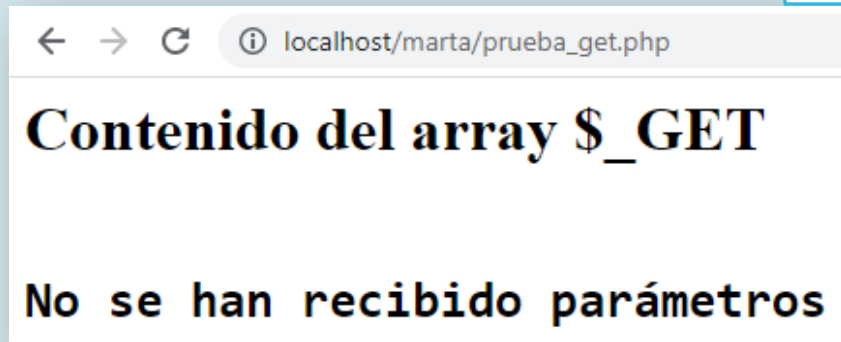


```
prueba_get.php •
prueba_get.php
1  <h1>Contenido del array $_GET</h1>
2  <pre>
3  <?php
4      print_r($_GET);
5  ?>
6  </pre>
```

Ejercicio 1:

\$_GET

1



2



Escribe un programa PHP que:

1. Si no recibe ningún parámetro en la URL, mostrará el mensaje:
No se han recibido parámetros
2. Si recibe algún parámetro en la URL, mostrará en forma de tabla los parámetros recibidos
3. Prueba el programa sin parámetros. Deberá mostrar la figura 1.
4. Prueba el programa con parámetros. Deberá mostrar la figura 2.

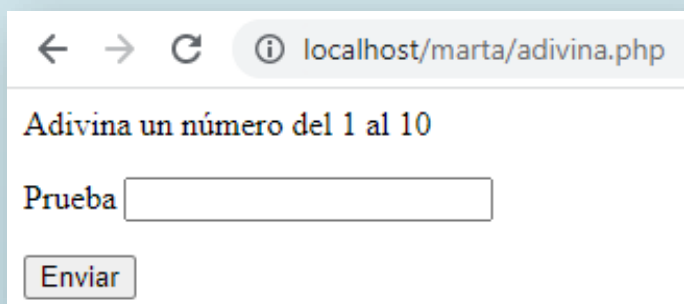
NOTA: Fíjate bien en la URL de ambas figuras

Formularios HTML: Submit

```
<html>
<head>
  <title>Ejemplo de formulario</title>
</head>
<body>
<p>Adivina un número del 1 al 10</p>
  <form>
    <p>
      <label for="prueba">Prueba</label>
      <input type="text" name="prueba" id="prueba"/>
    </p>
    <input type="submit"/>
  </form>
</body>
</html>
```

Ejemplo de formulario sencillo:

- Sólo hay un campo de entrada de datos, prueba.
- El form no indica action: Al hacer submit pedirá de nuevo la misma página
- El form no indica method: Se utilizará GET, método por defecto



← → ↻ ⓘ localhost/marta/adivina.php

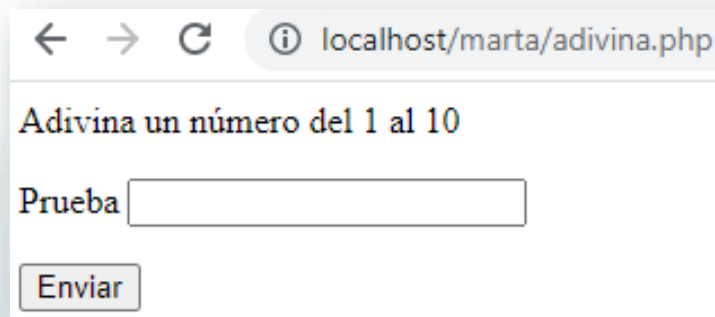
Adivina un número del 1 al 10

Prueba

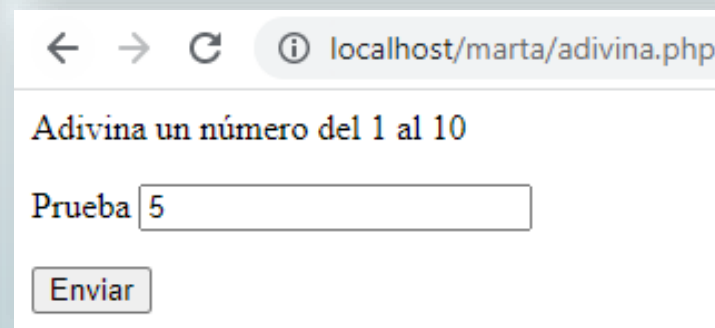
- Cuando hacemos un formulario en HTML, al pulsar el botón submit, el navegador realiza la acción indicada en el atributo action
- La acción normalmente es la petición de un fichero al servidor. Si no ponemos un action, se recargará la página.
- En el elemento **<form>** podemos indicar un atributo **method**, con valores "GET" o "POST":
 - Si ponemos **GET**, se enviarán los campos y sus variables como parámetros visibles en la URL
 - Si ponemos **POST**, se enviarán igualmente, pero no serán visibles en la URL
- Cada elemento **<input>** del form suele tener un atributo **name**, que equivale al nombre del parámetro que se enviará cuando se haga **submit**:
 - El nombre del parámetro es el valor del atributo **name**
 - El valor del parámetro es el contenido del campo
- El submit es un tipo de input que se visualiza en forma de botón. Al pulsarlo se realiza la acción.

Formularios HTML: Ejemplo de submit

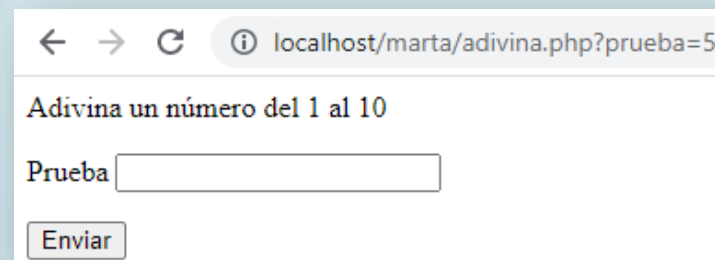
```
<html>
<head>
  <title>Ejemplo de formulario</title>
</head>
<body>
  <p>Adivina un número del 1 al 10</p>
  <form>
    <p>
      <label for="prueba">Prueba</label>
      <input type="text" name="prueba" id="prueba"/>
    </p>
    <input type="submit"/>
  </form>
</body>
</html>
```



A browser window with the address bar showing 'localhost/marta/adivina.php'. The page content is 'Adivina un número del 1 al 10'. Below this is a text input field labeled 'Prueba' and a submit button labeled 'Enviar'.



The browser window shows the same page, but the text input field now contains the number '5'. The submit button 'Enviar' remains below it.



The browser window shows the page after a submit action. The address bar now includes the parameter '?prueba=5'. The form content is identical to the previous state, with the text input field empty and the submit button 'Enviar' below it.

Primer paso:

- Pido la URL adivina.php sin parámetros
- El navegador carga la página
- Podremos escribir en el formulario

Segundo paso:

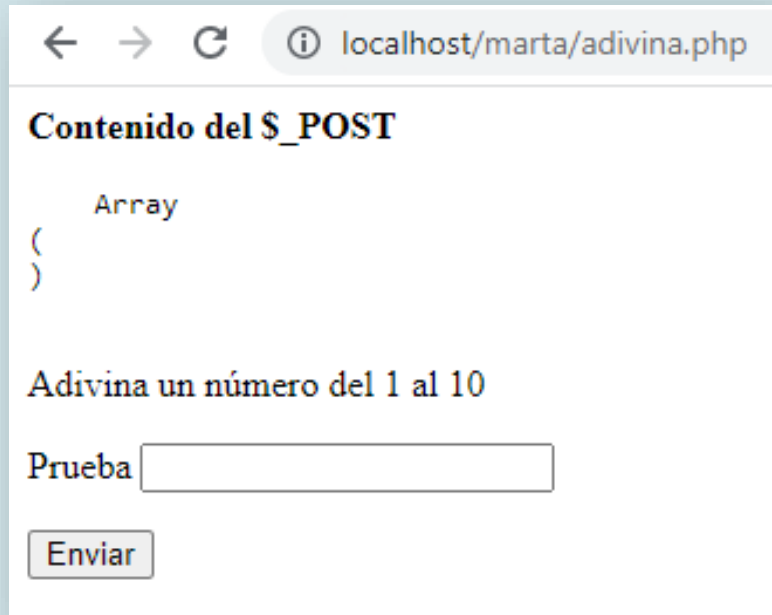
- Relleno los campos del formulario
- He puesto un 5 en el campo Prueba

Tercer paso:

- Hago submit
- El navegador vuelve a pedir adivina.php al servidor (no se indicó action)
- El navegador añade a la petición los parámetros (campos), en este caso prueba, de forma visible en la URL (no se indicó method, usa GET)

Ejercicio 2:

\$_POST



← → ↻ ⓘ localhost/marta/adivina.php

Contenido del \$_POST

Array
(
)

Adivina un número del 1 al 10

Prueba

Primera carga del formulario:

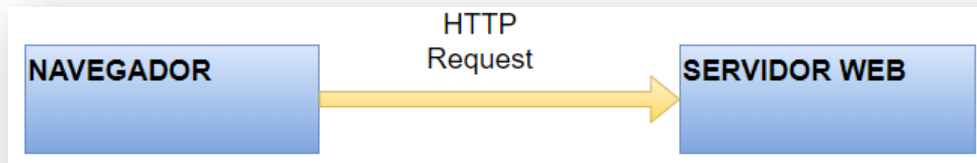
Array \$_POST vacío

Escribe el programa PHP de la página anterior.

1. Modifica el formulario para que el submit se realice con el método POST
2. Añade, dentro del body, pero antes del formulario, un fragmento de código PHP, para hacer un `print_r` (preformateado con `pre`) del array `$_POST`.
3. Observa lo que ocurre cuando ejecutas el programa la primera vez
4. Haz un submit con el formulario vacío y comprueba qué sucede
5. Haz un submit con un 5 en el campo prueba, y comprueba qué sucede

NOTA: Abre el inspector y comprueba todas las request que vas haciendo.

\$_GET vs \$_POST



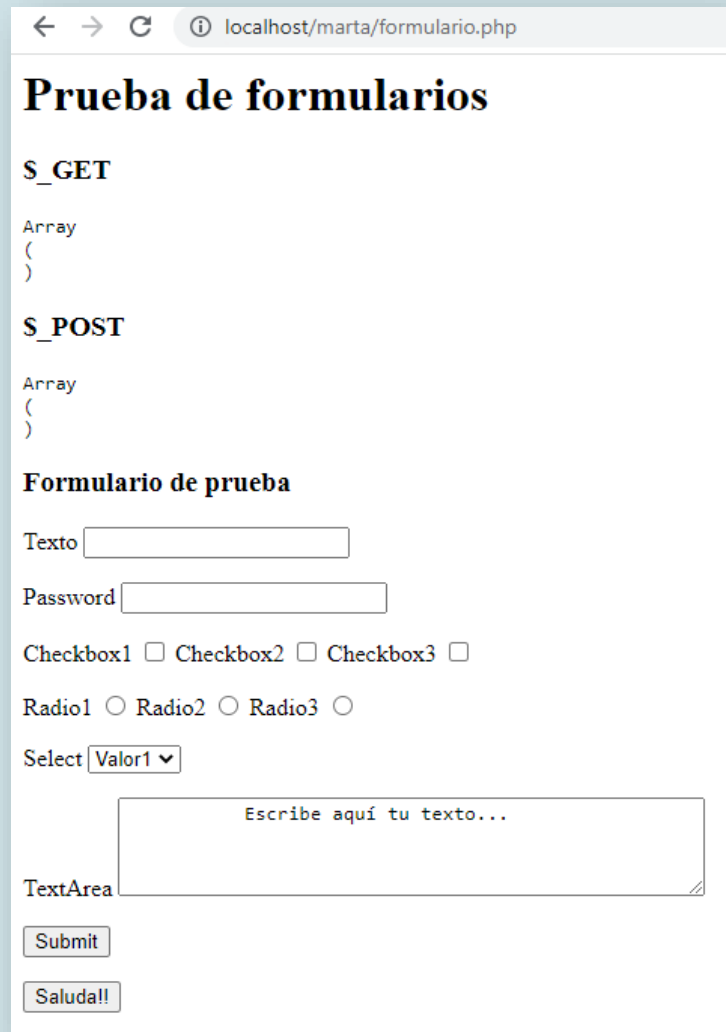
Name	× Headers Payload Preview Response Initiator Timing Cookies
adivina.php	<div>▼ General</div> <div>Request URL: http://localhost/marta/adivina.php</div> <div>Request Method: POST</div> <div>Status Code: 200 OK</div> <div>Remote Address: [::1]:80</div> <div>Referrer Policy: strict-origin-when-cross-origin</div>

Name	× Headers Payload Preview Response Initiator Timing Cookies
adivina.php?prueba=5	<div>▼ General</div> <div>Request URL: http://localhost/marta/adivina.php?prueba=5</div> <div>Request Method: GET</div> <div>Status Code: 200 OK</div> <div>Remote Address: [::1]:80</div> <div>Referrer Policy: strict-origin-when-cross-origin</div>

1. Con GET los datos viajan en la cabecera del mensaje Request
2. Suele utilizarse GET para **búsquedas**
3. Con POST los datos viajan en el cuerpo del mensaje Request
4. Suele utilizarse POST para **modificación** de información
5. Las **arañas** de la web (web crawlers o web spiders) suelen seguir los GET para indexar búsquedas, normalmente no siguen los POST
6. Una URL con parámetros GET debería obtener **siempre el mismo resultado**
7. Existe un **límite** en la cantidad de información que puede enviarse con GET (unos 3000 caracteres). Con POST no hay límite.
8. Con GET no pueden enviarse ficheros. Con POST sí.
9. Se recomienda utilizar GET en las búsquedas
10. Si vamos a enviar mucha información, se recomienda utilizar POST

Ejercicio 3:

Repaso de Formularios



← → ↻ ⓘ localhost/marta/formulario.php

Prueba de formularios

\$_GET

```
Array  
(  
)
```

\$_POST

```
Array  
(  
)
```

Formulario de prueba

Texto

Password

Checkbox1 ☐ Checkbox2 ☐ Checkbox3 ☐

Radio1 ☐ Radio2 ☐ Radio3 ☐

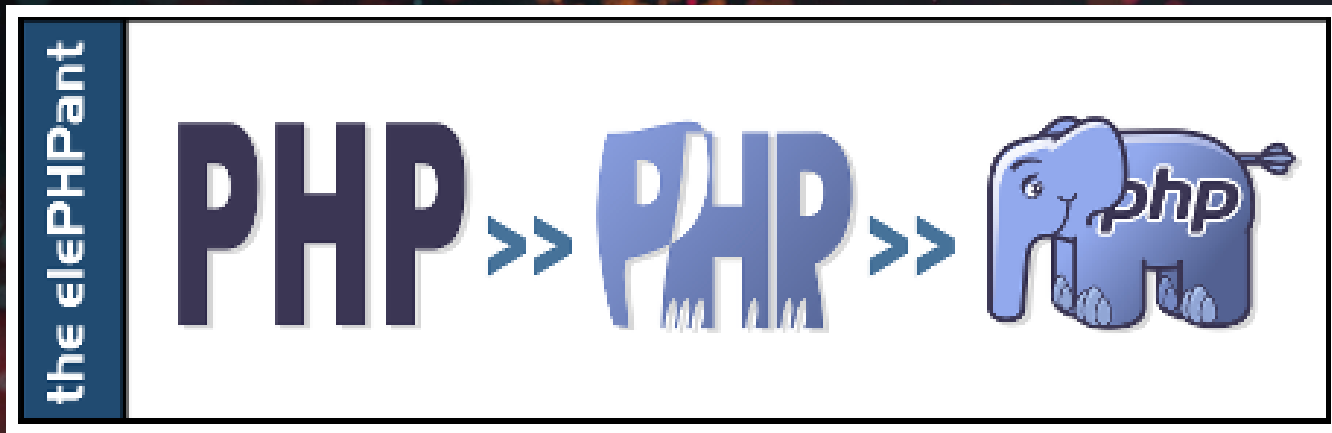
Select

TextArea

Vamos a repasar los formularios HTML para comprobar cómo envían la información al servidor

1. Crea un formulario con los varios tipos de input:
 1. Tipo Texto
 2. Tipo password
 3. Tres checkboxes
 4. Un radio button con tres opciones
 5. Un select-option con cuatro opciones
 6. Un submit con name y value
 7. Un textarea
 8. Un button con un onclick
 9. Un submit con nombre y valor
2. Antes del formulario, añade un bloque PHP para mostrar con `print_r` (preformateado con `pre`) el contenido de `$_GET` y `$_POST`
3. Prueba el submit del formulario con el método GET, y luego Pruébalo con el método POST, comprobando las request en el inspector

NOTA: El formulario tendrá el aspecto de la figura antes del submit.



<https://www.php.net/docs.php>