**HOW TO PASS VARIABLES AND DATA FROM PHP TO JAVASCRIPT**

There are actually several approaches to do this. Some require more overhead than others, and some are considered better than others.

In no particular order:

1. Use AJAX to get the data you need from the server.
2. Echo the data into the page somewhere, and use JavaScript to get the information from the DOM.
3. Echo the data directly to JavaScript.

In this post, we'll examine each of the above methods, and see the pros and cons of each, as well as how to implement them.

**1. Use AJAX to get the data you need from the server**

This method is considered the best, because **your server side and client side scripts are completely separate**.

Pros

* **Better separation between layers** - If tomorrow you stop using PHP, and want to move to a servlet, a REST API, or some other service, you don't have to change much of the JavaScript code.
* **More readable** - JavaScript is JavaScript, PHP is PHP. Without mixing the two, you get more readable code on both languages.
* **Allows for async data transfer** - Getting the information from PHP might be time/resources expensive. Sometimes you just don't want to wait for the information, load the page, and have the information reach whenever.
* **Data is not directly found on the markup** - This means that your markup is kept clean of any additional data, and only JavaScript sees it.

Cons

* **Latency** - AJAX creates an HTTP request, and HTTP requests are carried over network and have network latencies.
* **State** - Data fetched via a separate HTTP request won't include any information from the HTTP request that fetched the HTML document. You may need this information (e.g. if the HTML document is generated in response to a form submission) and, if you do, will have to transfer it across somehow. If you have ruled out embedding the data in the page (which you have if you are using this technique) then that limits you to cookies/sessions which may be subject to race conditions.

Implementation Example

With AJAX, you need two pages, one is where PHP generates the output, and the second is where JavaScript gets that output:

get-data.php

/\* Do some operation here, like talk to the database, the file-session

\* The world beyond, limbo, the city of shimmers, and Canada.

\*

\* AJAX generally uses strings, but you can output JSON, HTML and XML as well.

\* It all depends on the Content-type header that you send with your AJAX

\* request. \*/

echo json\_encode(42); //In the end, you need to **echo** the result.

//All data should be *json\_encode()*d.

//You can json\_encode() any value in PHP, arrays, strings,

//even objects.

index.php (or whatever the actual page is named like)

<!-- snip -->

<script>

function reqListener () {

console.log(this.responseText);

}

var oReq = new XMLHttpRequest(); //New request object

oReq.onload = function() {

//This is where you handle what to do with the response.

//The actual data is found on this.responseText

alert(this.responseText); //Will alert: 42

};

oReq.open("get", "get-data.php", true);

// ^ Don't block the rest of the execution.

// Don't wait until the request finishes to

// continue.

oReq.send();

</script>

<!-- snip -->

The above combination of the two files will alert 42 when the file finishes loading.

Some more reading material

* [**Using XMLHttpRequest - MDN**](https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest/Using_XMLHttpRequest)
* [**XMLHttpRequest object reference - MDN**](https://developer.mozilla.org/en/docs/Web/API/XMLHttpRequest)
* [**How to return the response from an Ajax call?**](http://stackoverflow.com/questions/14220321/how-to-return-the-response-from-an-ajax-call)

**2. Echo the data into the page somewhere, and use JavaScript to get the information from the DOM**

This method is less preferable to AJAX, but it still has its advantages. It's still *relatively* separated between PHP and JavaScript in a sense that there is no PHP directly in the JavaScript.

Pros

* **Fast** - DOM operations are often quick, and you can store and access a lot of data relatively quickly.

Cons

* **Potentially Unsemantic Markup** - Usually, what happens is that you use some sort of <input type=hidden> to store the information, because it's easier to get the information out of inputNode.value, but doing so means that you have a meaningless element in your HTML. HTML has the <meta> element for data about the document, and HTML 5 introduces data-\*attributes for data specifically for reading with JS that can be associated with particular elements.
* **Dirties up the Source** - Data that PHP generates is outputted directly to the HTML source, meaning that you get a bigger and less focused HTML source.
* **Harder to get structured data** - Structured data will have to be valid HTML, otherwise you'll have to escape and convert strings yourself.
* **Tightly couples PHP to your data logic** - Because PHP is used in presentation, you can't separate the two cleanly.

Implementation Example

With this, the idea is to create some sort of element which will not be displayed to the user, but is visible to JavaScript.

index.php

<!-- snip -->

<div id="dom-target" style="display: none;">

<?php

$output = "42"; //Again, do some operation, get the output.

echo htmlspecialchars($output); /\* You have to escape because the result

will not be valid HTML otherwise. \*/

?>

</div>

<script>

var div = document.getElementById("dom-target");

var myData = div.textContent;

</script>

<!-- snip -->

**3. Echo the data directly to JavaScript**

This is probably the easiest to understand, and the most horrible to use. Don't do this unless you know what you're doing.

Pros

* **Very easily implemented** - It takes very little to implement this, and understand.
* **Does not dirty source** - Variables are outputted directly to JavaScript, so the DOM is not affected.

Cons

* ***Insecure*** - PHP has no trivial JavaScript escape functions, and they aren't trivial to implement. Especially when using user inputs, you are **extremely** vulnerable to second tier injections.**Disputed** see comments
* **Tightly couples PHP to your data logic** - Because PHP is used in presentation, you can't separate the two cleanly.
* **Structured data is hard** - You can probably do JSON... kinda. But XML and HTML will require special attention.

Implementation Example

Implementation is relatively straightforward:

<!-- snip -->

<script>

var data = <?php echo json\_encode("42"); ?>; //Don't forget the extra semicolon!

</script>

<!-- snip -->

Good luck!