JavaScript Performance

❮ Previous Next ❯

How to speed up your JavaScript code.

Reduce Activity in Loops

Loops are often used in programming.

Each statement in a loop, including the for statement, is executed for each iteration of the loop.

Statements or assignments that can be placed outside the loop will make the loop run faster.

Bad:

```
var i;
for (i = 0; i < arr.length; i++) {
```

Better Code:

```
var i;
var l = arr.length;
for (i = 0; i < l; i++) {
```

The bad code accesses the length property of an array each time the loop is iterated.

The better code accesses the length property outside the loop and makes the loop run faster.

Reduce DOM Access

Accessing the HTML DOM is very slow, compared to other JavaScript statements.

If you expect to access a DOM element several times, access it once, and use it as a local variable:

Example

```
var obj;
obj = document.getElementById("demo");
obj.innerHTML = "Hello";
```

Try it Yourself »

Learn How To Create an Image Slideshow

Reduce DOM Size

Keep the number of elements in the HTML DOM small.

This will always improve page loading, and speed up rendering (page display), especially on smaller devices.

Every attempt to search the DOM (like getElementsByTagName) will benefit from a smaller DOM.

Avoid Unnecessary Variables

Don't create new variables if you don't plan to save values.

Often you can replace code like this:

```
var fullName = firstName + " " + lastName;
document.getElementById("demo").innerHTML = fullName;
```

With this:

```
document.getElementById("demo").innerHTML = firstName + " " + lastName
```

Delay JavaScript Loading

Putting your scripts at the bottom of the page body lets the browser load the page first.

While a script is downloading, the browser will not start any other downloads. In addition all parsing and rendering activity might be blocked.

> The HTTP specification defines that browsers should not download more than two components in parallel.

An alternative is to use **defer="true"** in the script tag. The defer attribute specifies that the script should be executed after the page has finished parsing, but it only works for external scripts.

If possible, you can add your script to the page by code, after the page has loaded:

Example

```
<script>
window.onload = function() {
    var element = document.createElement("script");
    element.src = "myScript.js";
    document.body.appendChild(element);
};
</script>
```

Avoid Using with

Avoid using the **with keyword**. It has a negative effect on speed. It also clutters up JavaScript scopes.

The with keyword is **not allowed** in strict mode.

❮ Previous Next ❯