24th September 2019

Hello, world!

This part of the tutorial is about core JavaScript, the language itself.

But we need a working environment to run our scripts and, since this book is online, the browser is a good choice. We'll keep the amount of browser-specific commands (like `alert`) to a minimum so that you don't spend time on them if you plan to concentrate on another environment (like Node.js). We'll focus on JavaScript in the browser in the next part of the tutorial.

So first, let's see how we attach a script to a webpage. For server-side environments (like Node.js), you can execute the script with a command like `"node my.js"`.

The "script" tag

JavaScript programs can be inserted into any part of an HTML document with the help of the `<script>` tag.

For instance:

```
<!DOCTYPE HTML>
<html>

<body>

  <p>Before the script...</p>

  <script>
    alert( 'Hello, world!' );
  </script>

  <p>...After the script.</p>

</body>

</html>
```

You can run the example by clicking the "Play" button in the right-top corner of the box above.

The `<script>` tag contains JavaScript code which is automatically executed when the browser processes the tag.

Modern markup

The `<script>` tag has a few attributes that are rarely used nowadays but can still be found in old code:

The `type` attribute: `<script type=…>`

The old HTML standard, HTML4, required a script to have a `type`. Usually it was `type="text/javascript"`. It's not required anymore. Also, the modern HTML standard totally changed the meaning of this attribute. Now, it can be used for JavaScript modules. But that's an advanced topic; we'll talk about modules in another part of the tutorial.

The `language` attribute: `<script language=…>`

This attribute was meant to show the language of the script. This attribute no longer makes sense because JavaScript is the default language. There is no need to use it.

Comments before and after scripts.

In really ancient books and guides, you may find comments inside `<script>` tags, like this:

```
<script type="text/javascript"><!--
    ...
//--></script>
```

This trick isn't used in modern JavaScript. These comments hide JavaScript code from old browsers that didn't know how to process the `<script>` tag. Since browsers released in the last 15 years don't have this issue, this kind of comment can help you identify really old code.

External scripts

If we have a lot of JavaScript code, we can put it into a separate file.

Script files are attached to HTML with the `src` attribute:

```
<script src="/path/to/script.js"></script>
```

Here, `/path/to/script.js` is an absolute path to the script from the site root. One can also provide a relative path from the current page. For instance, `src="script.js"` would mean a file `"script.js"` in the current folder.

We can give a full URL as well. For instance:

```
<script src="https://cdnjs.cloudflare.com/ajax/libs/lodash.js/3.2.0/lodash.js
```

To attach several scripts, use multiple tags:

```
<script src="/js/script1.js"></script>
<script src="/js/script2.js"></script>
…
```

Please note:

As a rule, only the simplest scripts are put into HTML. More complex ones reside in separate files.

The benefit of a separate file is that the browser will download it and store it in its cache.

Other pages that reference the same script will take it from the cache instead of downloading it, so the file is actually downloaded only once.

That reduces traffic and makes pages faster.

If `src` is set, the script content is ignored.

A single `<script>` tag can't have both the `src` attribute and code inside.

This won't work:

```
<script src="file.js">
  alert(1); // the content is ignored, because src is set
</script>
```

We must choose either an external `<script src="…">` or a regular `<script>` with code.

The example above can be split into two scripts to work:

```
<script src="file.js"></script>
<script>
  alert(1);
</script>
```

Summary

- We can use a `<script>` tag to add JavaScript code to a page.
- The `type` and `language` attributes are not required.
- A script in an external file can be inserted with `<script src="path/to/script.js"></script>` .

There is much more to learn about browser scripts and their interaction with the webpage. But let's keep in mind that this part of the tutorial is devoted to the JavaScript language, so we shouldn't distract ourselves with browser-specific implementations of it. We'll be using the browser as a way to run JavaScript, which is very convenient for online reading, but only one of many.

Tasks

Show an alert

importance: 5

Create a page that shows a message “I’m JavaScript!”.

Do it in a sandbox, or on your hard drive, doesn’t matter, just ensure that it works.

Demo in new window

solution

Show an alert with an external script

importance: 5

Take the solution of the previous task Show an alert. Modify it by extracting the script content into an external file `alert.js`, residing in the same folder.

Open the page, ensure that the alert works.

solution

Previous lesson Next lesson

Share

Tutorial map

Comments

- If you have suggestions what to improve - please submit a GitHub issue or a pull request instead of commenting.
- If you can't understand something in the article – please elaborate.
- To insert a few words of code, use the `<code>` tag, for several lines – use `<pre>`, for more than 10 lines – use a sandbox (plnkr, JSBin, codepen…)