Hello, world!

The tutorial that you're reading is about core JavaScript, which is platform-independent. We need a working environment to run our scripts. The browser is a good choice.

The "script" tag

JavaScript programs can be inserted in any place of HTML 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>
```

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

External scripts

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

The script file is attached to HTML with `src` attribute:

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

Here `/path/to/script.js` is an absolute path to the file with the script (from the site root).

It is also possible to provide a path relative to the current page. For instance, `src="script.js"` would mean a file `"script.js"` from 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"></script>
```

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 then store in its cache.

After this, other pages which want the same script will take it from the cache instead of downloading it. So the file is actually downloaded only once.

That saves traffic and makes pages faster.

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

A single `<script>` tag may not have both `src` attribute and the 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 it's 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>
```

Debugging

Often you'll come across a program that doesn't work. This could be due to one of two problems: 1. Syntax error or 2. Logic error. Syntax errors are errors in syntax i.e. spelling, leaving out commas, semi-colons etc. They prevent a program from running. One way to find these errors is with the help of your browser's debugger. Press CTRL SHIFT I to bring up the debugger. Note that you should be logged out of any online accounts (i.e. google account) before doing this.

Logic errors are sometimes more difficult to find as they will run but not perform as expected. For example you ask a user for two numbers to show their sum when in fact you show their product. In this case it becomes necessary to trace the values of your variables. This will be discussed in more depth later on.

Summary

- We can use a `<script>` tag to add JavaScript code to the page.
- A script in an external file can be inserted with `<script src="path/to/script.js"></script>`.
- alert(message); is a simple statement that displays a messagebox with a message as its parameter
- debugging is the process of finding errors in your code
- there are two main types of errors: syntax and logic
- syntax are due to errors due to spelling or not writing the code correctly and prevent your program from executing
- logic errors do not prevent program execution but prevent a program from properly solving its problem

Questions and Exercises

1. Create a page that shows a message **"I'm JavaScript**!" using an alert box **(*introToJS_1.html)**.
2. Make a **JS** program using an **external script**. In the external script add several alert statements explaining how to include **JS** into a web page. **(introToJS_2.html, introToJS_2.js)**
3. What is the file extension on a file containing **JS** only (answer all question using **red font**)?
    - Java Script
4. Create a new web page referencing an external **JS** file and write a program to display a poem in an alert box. Use **\n** to add line breaks in your message. For example alert("Hello \n there!"); should show up as
   Hello
   there!
   (introJS_4.html, introToJS_4.js)
5. Create a new program. Using alert boxes explain the main types of errors found in a program. Intentionally create one specific syntax error on each line. Run the debugger and see how well it helps you find those errors. **(introToJS_5.html)**

6. Do some research and list and describe all the different ways to bring up the debugger and console window.

Do not forget to link the exercises and this handout to your class website. You are responsible for maintaining all your work on your site regularly. At the end of the semester your site will be evaluated.

***From now on name your project files based on the handout and exercise #. Examples will be provided on this handout only.**