Chapter 4: Comments

Section 4.1: Using Comments

To add annotations, hints, or exclude some code from being executed JavaScript provides two ways of commenting code lines

Single line Comment `//`

Everything after the `//` until the end of the line is excluded from execution.

```
function elementAt( event ) {
  // Gets the element from Event coordinates
  return document.elementFromPoint(event.clientX, event.clientY);
}
// TODO: write more cool stuff!
```

Multi-line Comment `/**/`

Everything between the opening `/*` and the closing `*/` is excluded from execution, even if the opening and closing are on different lines.

```
/*
   Gets the element from Event coordinates.
   Use like:
   var clickedEl = someEl.addEventListener("click", elementAt, false);
*/
function elementAt( event ) {
  return document.elementFromPoint(event.clientX, event.clientY);
}
/* TODO: write more useful comments! */
```

Section 4.2: Using HTML comments in JavaScript (Bad practice)

HTML comments (optionally preceded by whitespace) will cause code (on the same line) to be ignored by the browser also, though this is considered **bad practice**.

One-line comments with the HTML comment opening sequence (`<!--`):

> **Note:** the JavaScript interpreter ignores the closing characters of HTML comments (`-->`) here.

```
<!-- A single-line comment.
<!-- --> Identical to using `//` since
<!-- --> the closing `-->` is ignored.
```

This technique can be observed in legacy code to hide JavaScript from browsers that didn't support it:

```
<script type="text/javascript" language="JavaScript">
<!--
/* Arbitrary JavaScript code.
   Old browsers would treat
   it as HTML code. */
// -->
```

```
</script>
```

An HTML closing comment can also be used in JavaScript (independent of an opening comment) at the beginning of a line (optionally preceded by whitespace) in which case it too causes the rest of the line to be ignored:

```
--> Unreachable JS code
```

These facts have also been exploited to allow a page to call itself first as HTML and secondly as JavaScript. For example:

```
<!--
self.postMessage('reached JS "file"');
/*
-->
<!DOCTYPE html>
<script>
var w1 = new Worker('#1');
w1.onmessage = function (e) {
    console.log(e.data); // 'reached JS "file"
};
</script>
<!--
*/
-->
```

When run a HTML, all the multiline text between the `<!--` and `-->` comments are ignored, so the JavaScript contained therein is ignored when run as HTML.

As JavaScript, however, while the lines beginning with `<!--` and `-->` are ignored, their effect is not to escape over *multiple* lines, so the lines following them (e.g., `self.postMessage(...)` will not be ignored when run as JavaScript, at least until they reach a *JavaScript* comment, marked by `/*` and `*/`. Such JavaScript comments are used in the above example to ignore the remaining *HTML* text (until the `-->` which is also ignored as JavaScript).