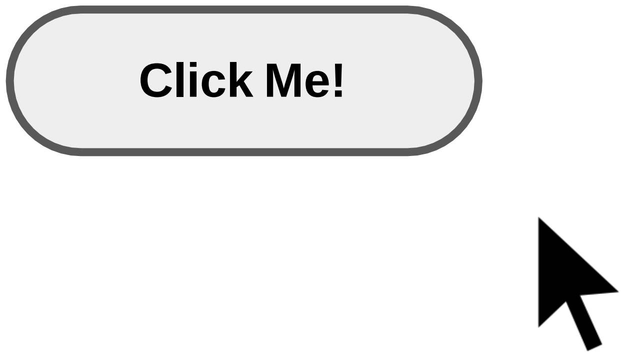
DHTML and Events

Event-driven programming

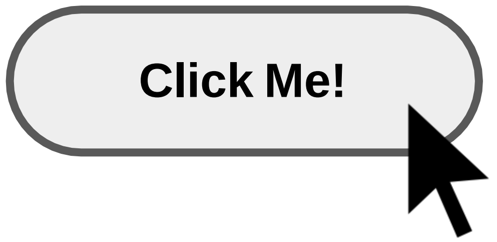
Most JavaScript written in the browser is **event-driven**:
The code doesn't run right away, but it executes after some event fires.



Example:
Here is a UI element that the user can interact with.

Event-driven programming

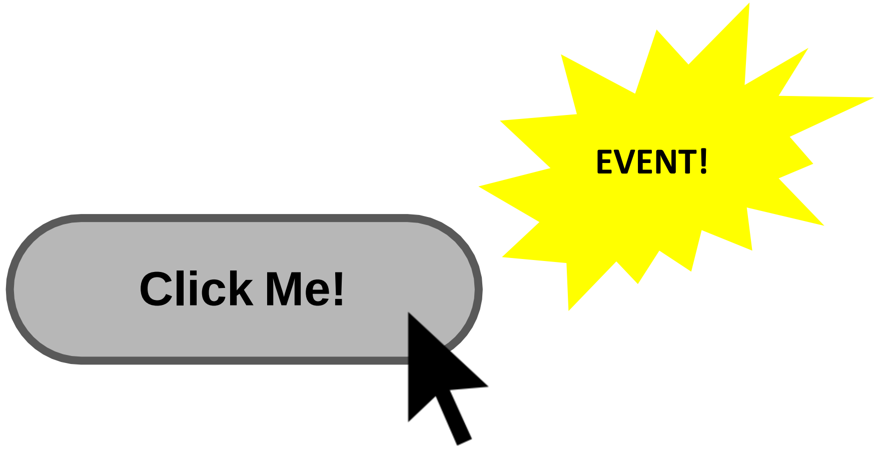
Most JavaScript written in the browser is **event-driven**:
The code doesn't run right away, but it executes after some event fires.



When the user clicks the button...

Event-driven programming

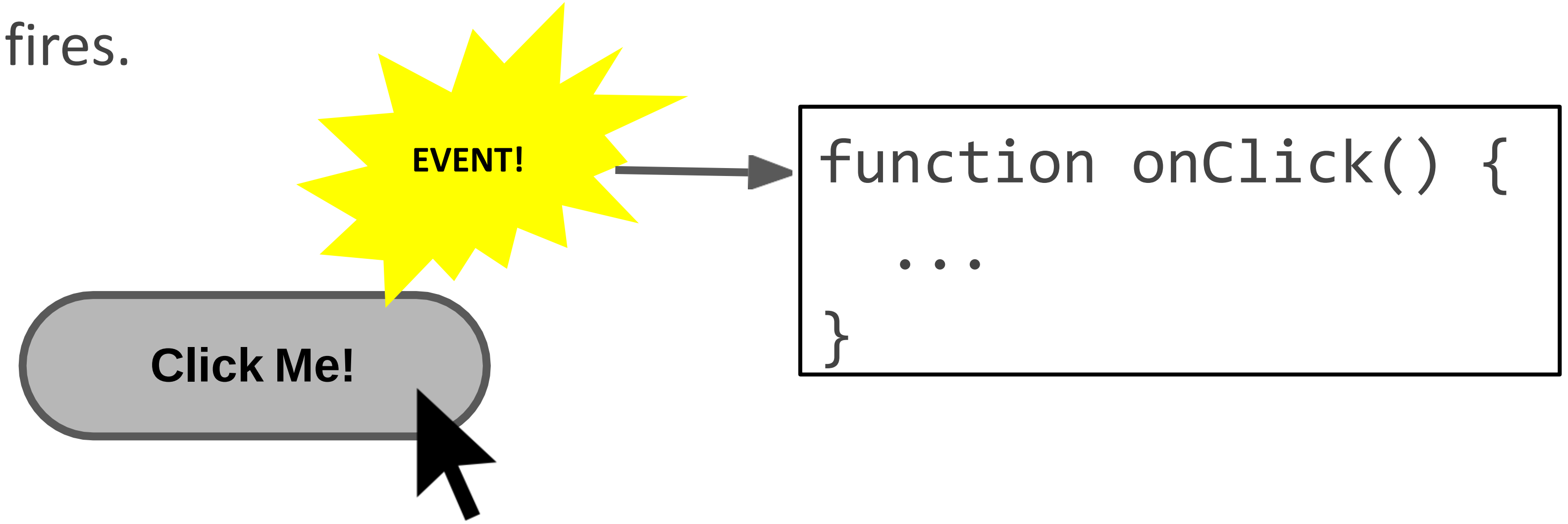
Most JavaScript written in the browser is **event-driven**:
The code doesn't run right away, but it executes after some event fires.



...the button emits an "**event**," which is like an announcement that some interesting thing has occurred.

Event-driven programming

Most JavaScript written in the browser is **event-driven**:
The code doesn't run right away, but it executes after some event fires.



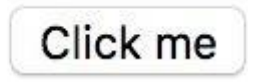
```
function onClick() {
  ...
}
```

Any function listening to that event now executes. This function is called an "**event handler**."

A few more HTML elements

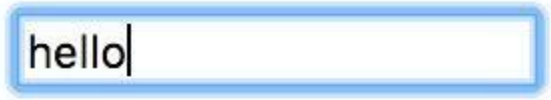
Buttons:

```
<button>Click me</button>
```



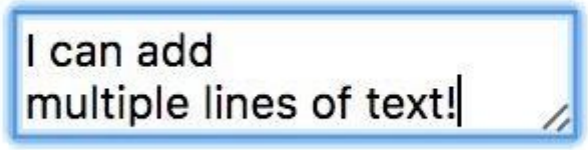
Single-line text input:

```
<input type="text" />
```



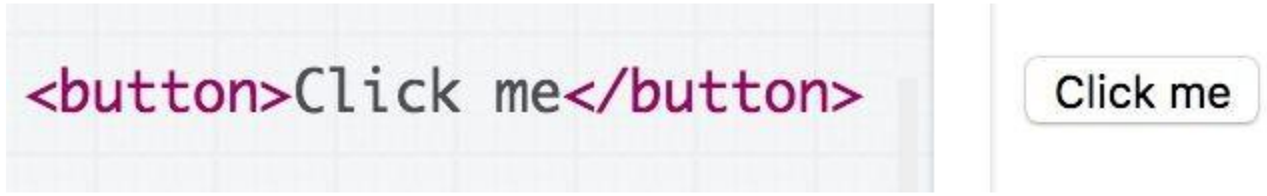
Multi-line text input:

```
<textarea></textarea>
```



Using event listeners

Let's print "Clicked" to the Web Console when the user clicks the given button:



```
<button>Click me</button>
```

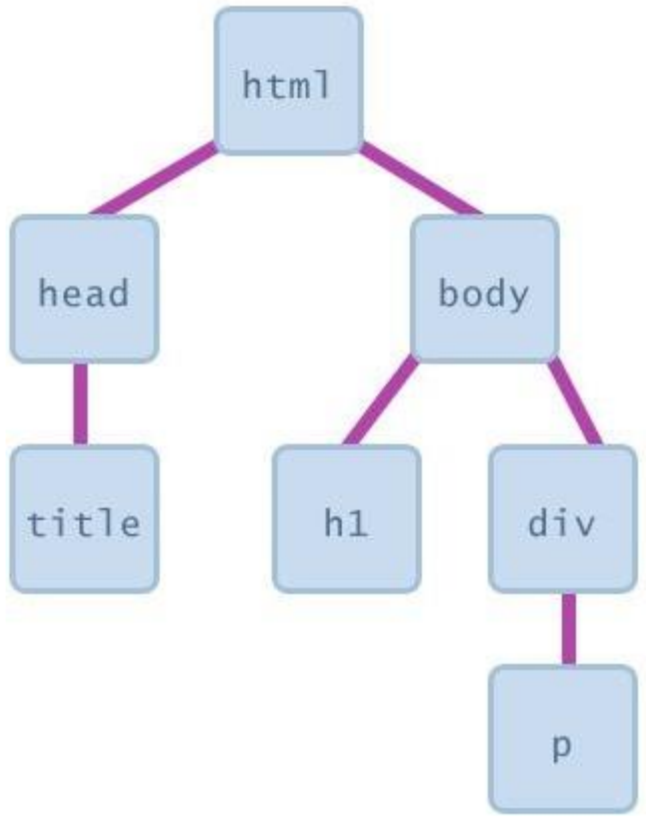
We need to add an event listener to the button...

How do we talk to an element in HTML from JavaScript?

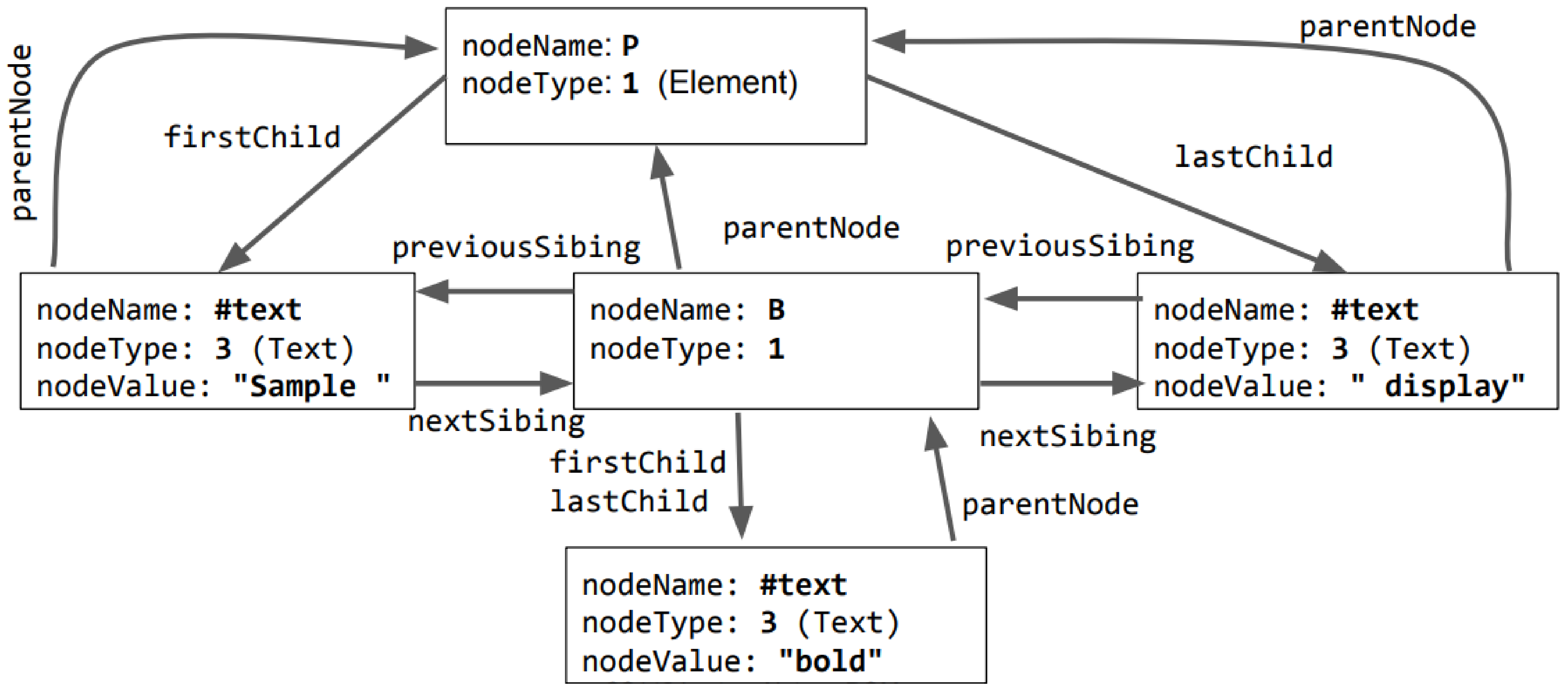
The DOM

Every element on a page is accessible in JavaScript through the **DOM**: **Document Object Model**

- The DOM is the tree of nodes corresponding to HTML elements on a page.
- Can modify, add and remove nodes on the DOM, which will modify, add, or remove the corresponding element on the page.



```
<p>Sample <b>bold</b> display</p>
```



Accessing DOM Nodes

- Walk DOM hierarchy (not recommended)

  ```
  element = document.body.firstChild.nextSibling.firstChild;
  element.setAttribute(…
  ```

- Use DOM lookup method. An example using ids:

  HTML: `<div id="div42">...</div>`

  ```
  element = document.getElementById("div42");
  element.setAttribute(…
  ```

- Many: `getElementsByClassName(), getElementsByTagName(), …`
  - Can start lookup at any element:

    `document.body.firstChild.getElementsByTagName()`

Getting DOM objects

We can access an HTML element's corresponding DOM object in JavaScript via the `querySelector` function:

`document.querySelector('`***css selector***`');`

- This returns the **first** element that matches the given CSS selector

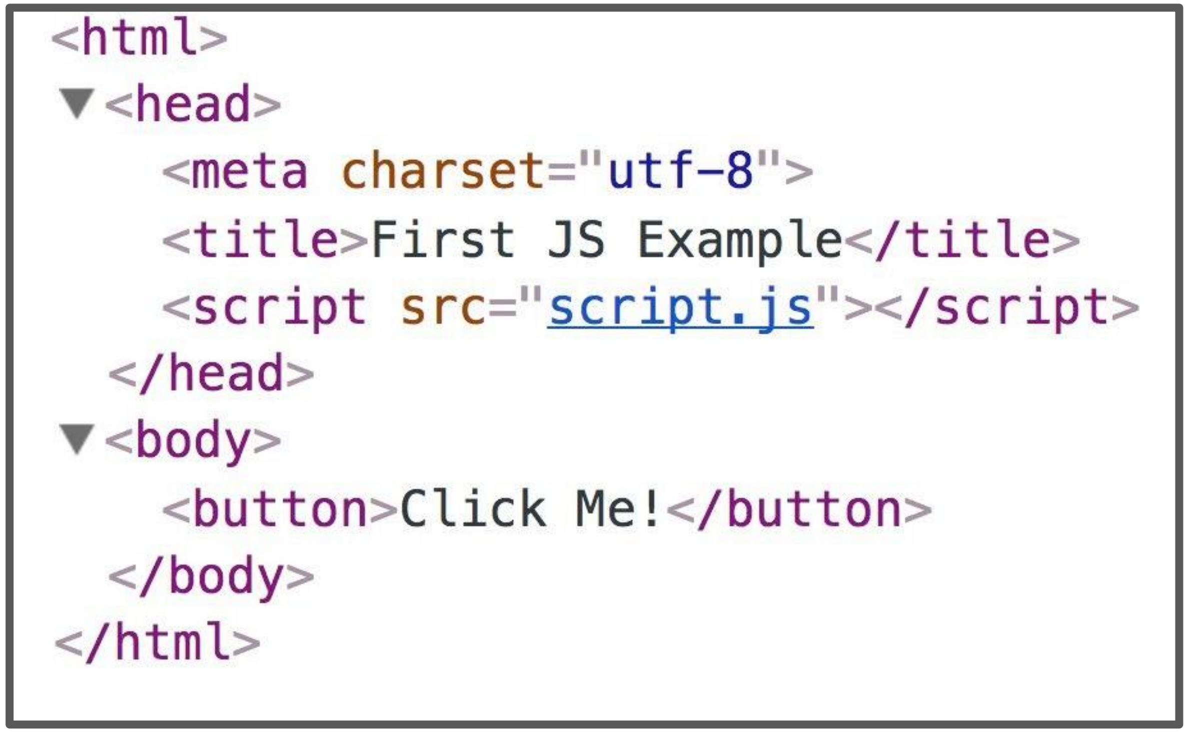
```
// Returns the element with id="button"
let element = document.querySelector('#button');
```

Adding event listeners

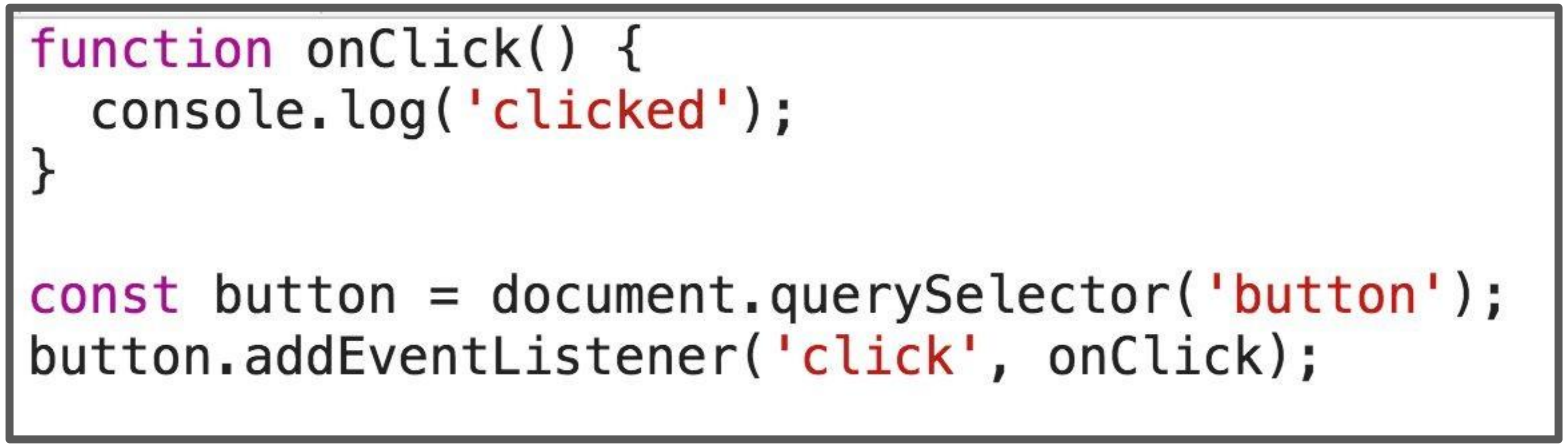
Each DOM object has the following function:

`addEventListener(`***event name, function name***`);`

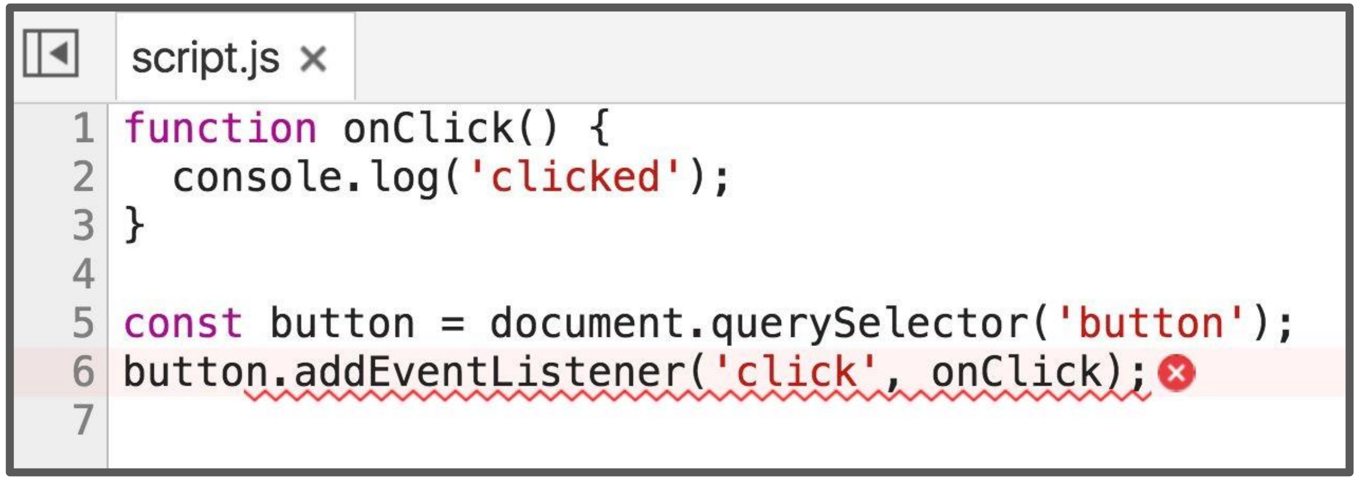
- ***event name*** is the string name of the JavaScript event you want to listen to
  - Common ones: click, focus, blur, etc
- ***function name*** is the name of the JavaScript function you want to execute when the event fires



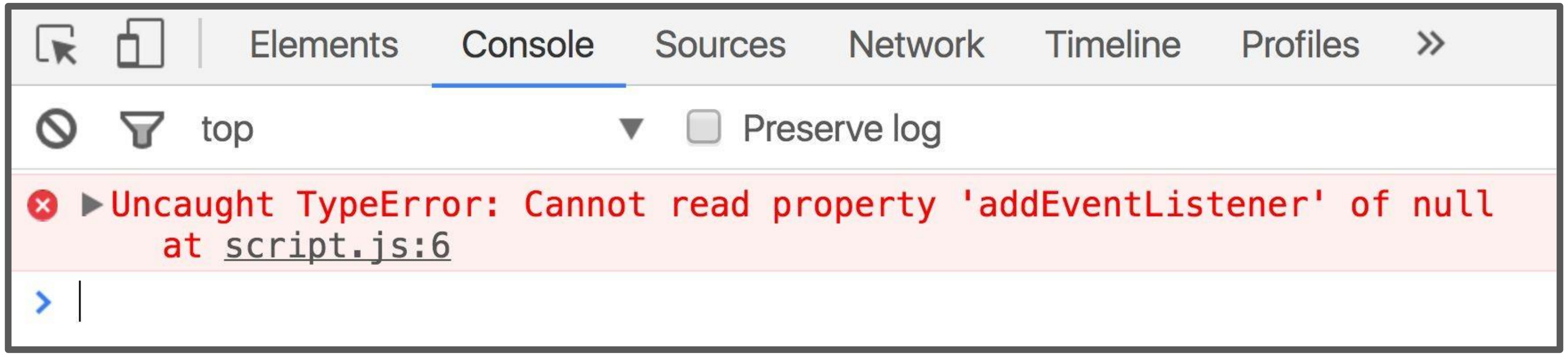
```
<html>
▼<head>
    <meta charset="utf-8">
    <title>First JS Example</title>
    <script src="script.js"></script>
  </head>
▼<body>
    <button>Click Me!</button>
  </body>
</html>
```



```
function onClick() {
  console.log('clicked');
}

const button = document.querySelector('button');
button.addEventListener('click', onClick);
```



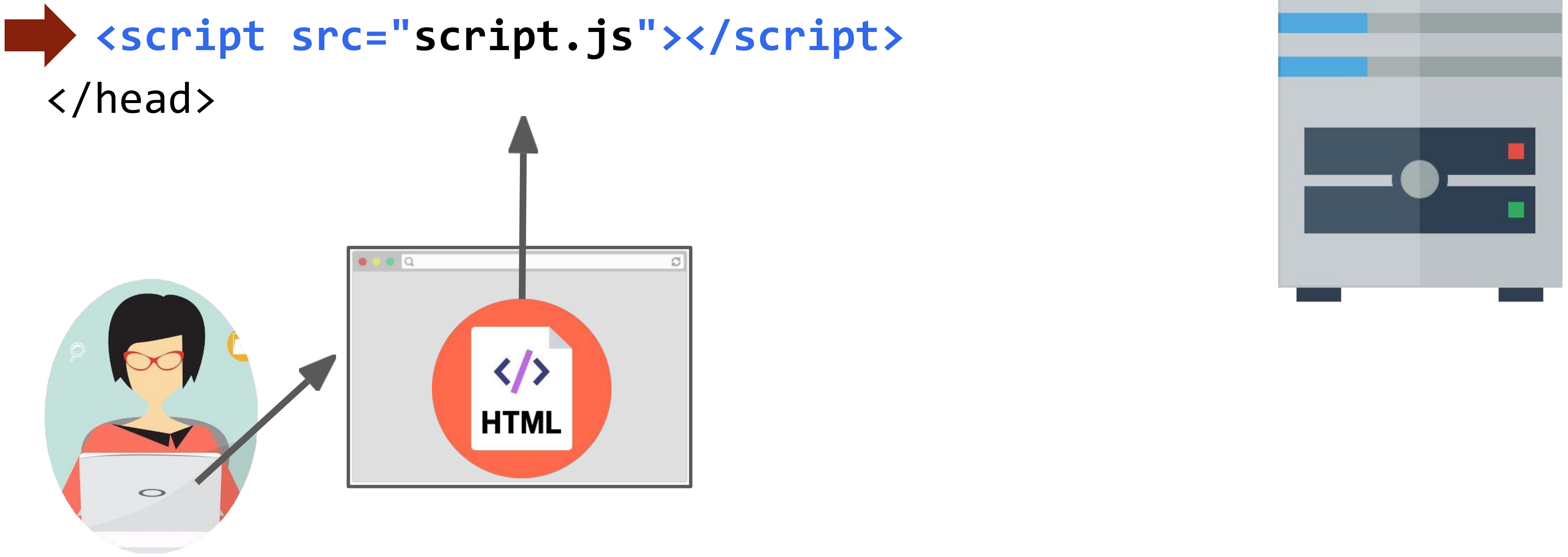
```
function onClick() {
  console.log('clicked');
}

const button = document.querySelector('button');
button.addEventListener('click', onClick);
```



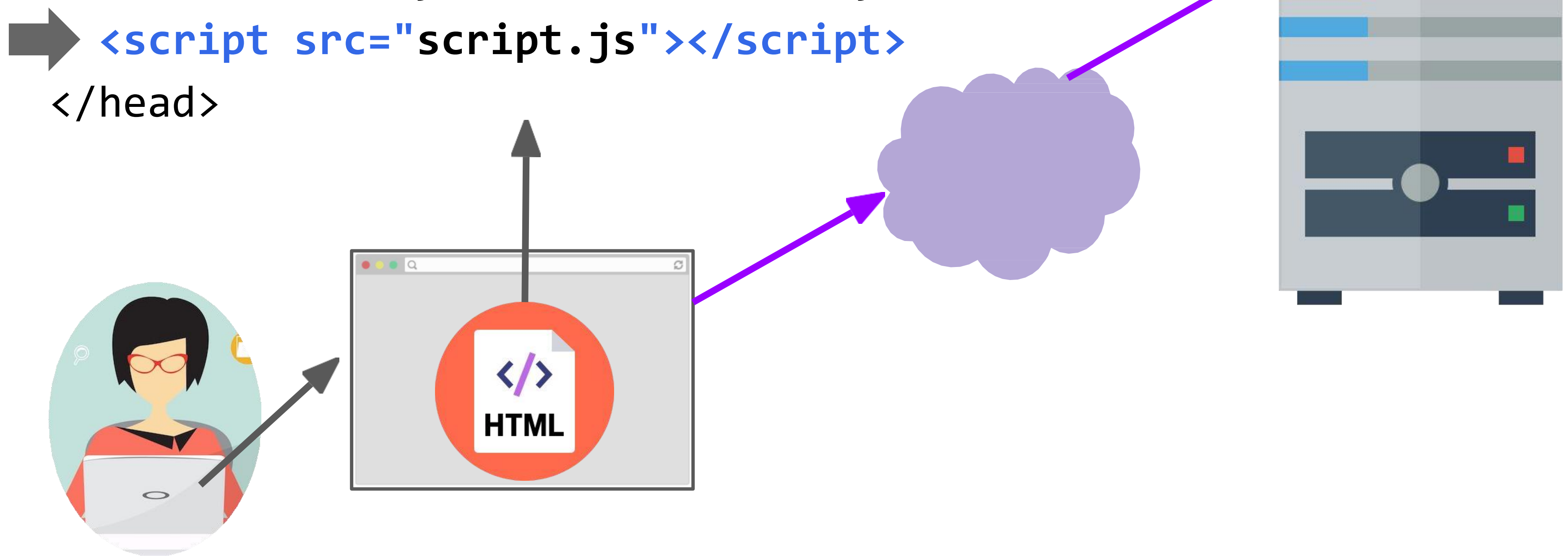
```
Uncaught TypeError: Cannot read property 'addEventListener' of null
    at script.js:6
```

Error! Why?

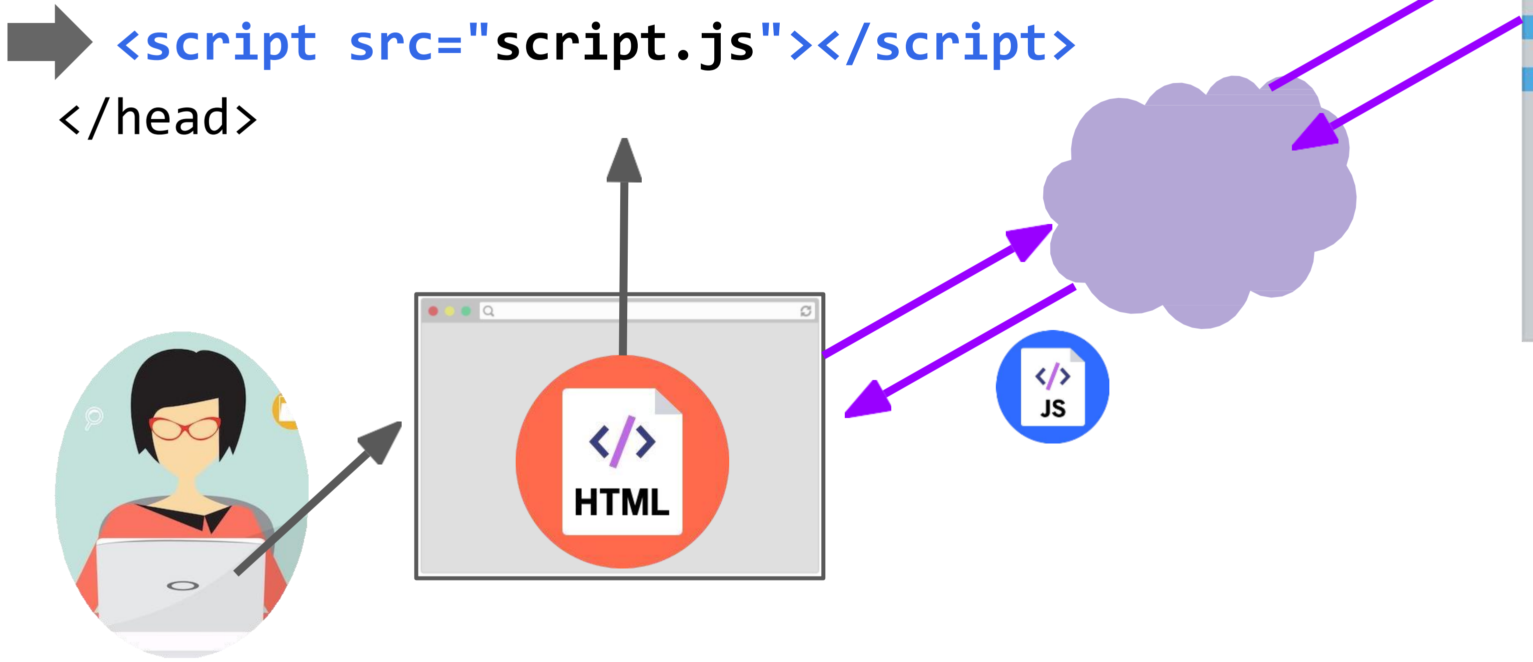
```
<head>
  <title>CS 353</title>
  <link rel="stylesheet" href="style.css" />
  <script src="script.js"></script>
</head>
```

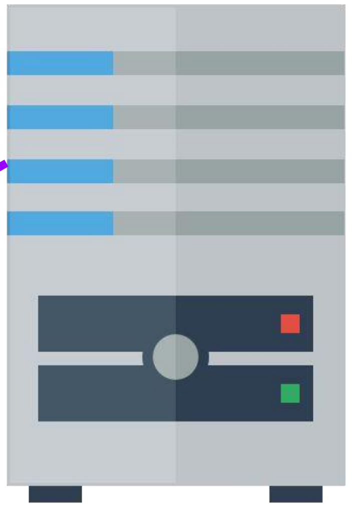


```
<head>
  <title>CS 353</title>
  <link rel="stylesheet" href="style.css" />
  <script src="script.js"></script>
</head>
```

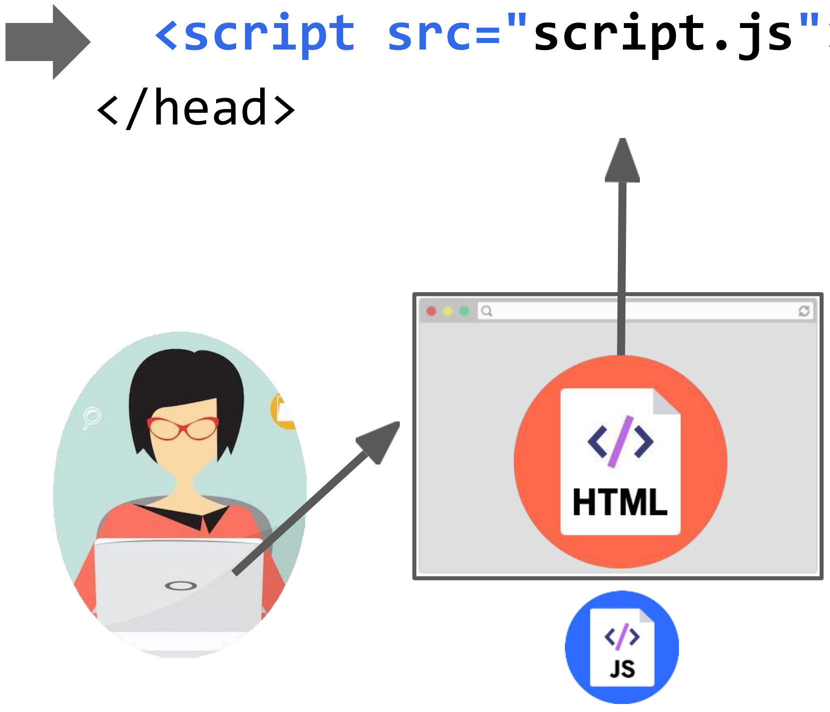


```
<head>
  <title>CS 353</title>
  <link rel="stylesheet" href="style.css" />
  <script src="script.js"></script>
</head>
```

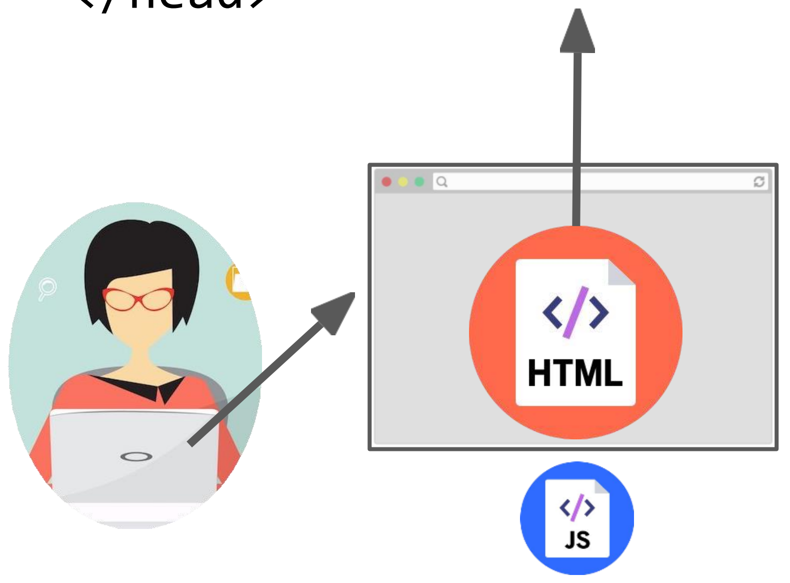




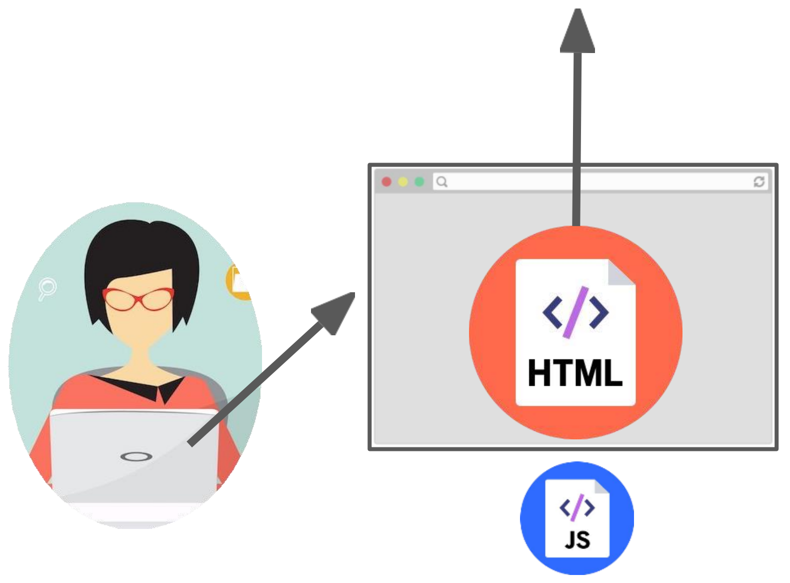
```
<head>
  <title>CS 353</title>
  <link rel="stylesheet" href="style.css" />
  <script src="script.js"></script>
</head>
```



```
function onClick() {
  console.log('clicked');
}

const button = document.querySelec
button.addEventListener('click', o
<head>
  <title>CS 353</title>
  <link rel="stylesheet" href="style.css" />
  <script src="script.js"></script>
</head>
```



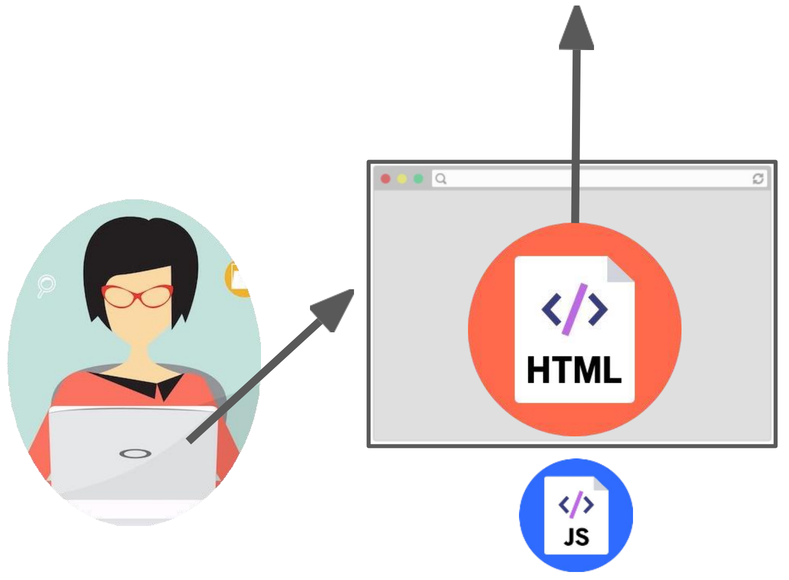
```
function onClick() {
  console.log('clicked');
}

const button = document.querySelec
button.addEventListener('click', o
<head>
  <title>CS 353</title>
  <link rel="stylesheet" href="style.css" />
  <script src="script.js"></script>
</head>
```



```
function onClick() {
  console.log('clicked');
}

const button = document.querySelec
button.addEventListener('click', o
```

We are only at the `<script>` tag, which is at the top of the document... so the `<button>` isn't available yet.

```
<head>
  <title>CS 353</title>
  <link rel="stylesheet" href="style.css" />
  <script src="script.js"></script>
</head>
```



```
function onClick() {
  console.log('clicked');
}

const button = document.querySelec
button.addEventListener('click', o
```

Therefore `querySelector` returns `null`, and we can't call `addEventListener` on `null`.

Use `defer`

You can add the `defer` attribute onto the script tag so that the JavaScript doesn't execute until after the DOM is loaded (mdn):

```
<script src="script.js" defer></script>
```

Use `defer`

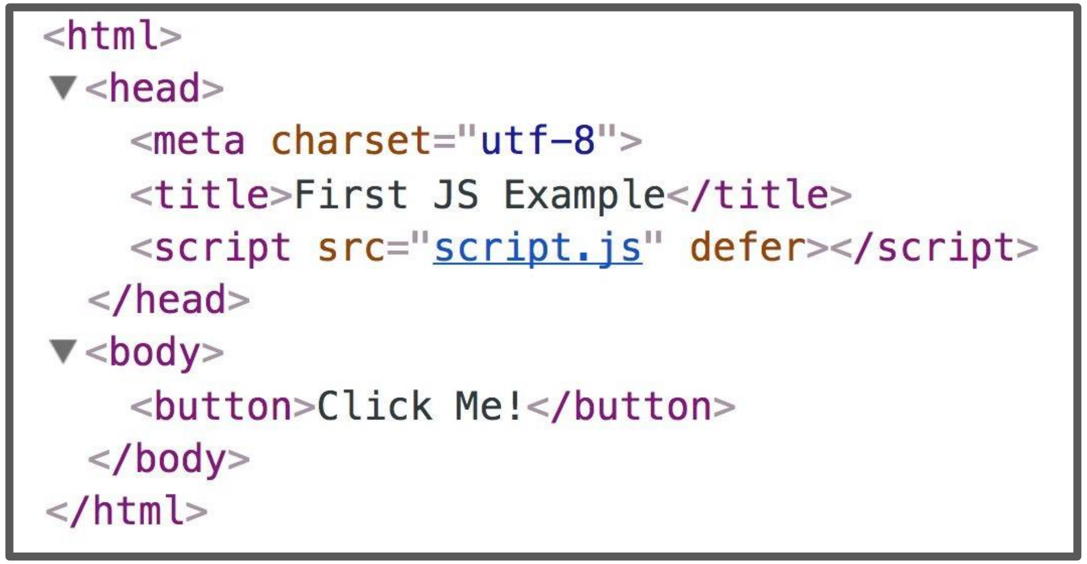
You can add the `defer` attribute onto the script tag so that the JavaScript doesn't execute until after the DOM is loaded (mdn):

```
<script src="script.js" defer></script>
```

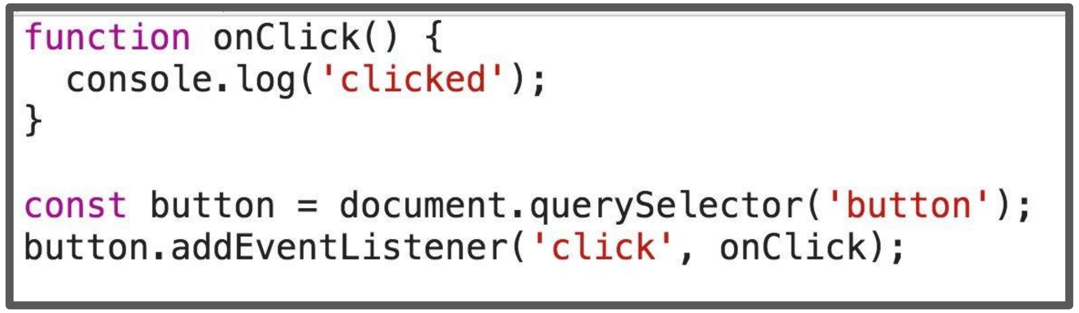
Other old-school ways of doing this (**don't do these**):

- Put the `<script>` tag at the bottom of the page
- Listen for the "`load`" event on the window object

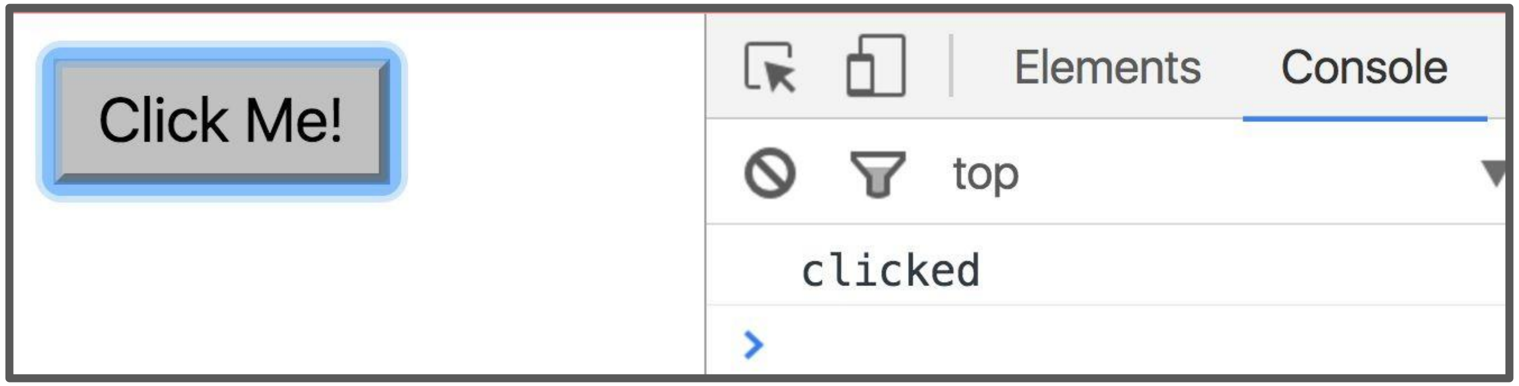
You will see tons of examples on the internet that do this. They are out of date. defer is widely supported and better.



```
<html>
▼<head>
    <meta charset="utf-8">
    <title>First JS Example</title>
    <script src="script.js" defer></script>
  </head>
▼<body>
    <button>Click Me!</button>
  </body>
</html>
```



```
function onClick() {
  console.log('clicked');
}

const button = document.querySelector('button');
button.addEventListener('click', onClick);
```



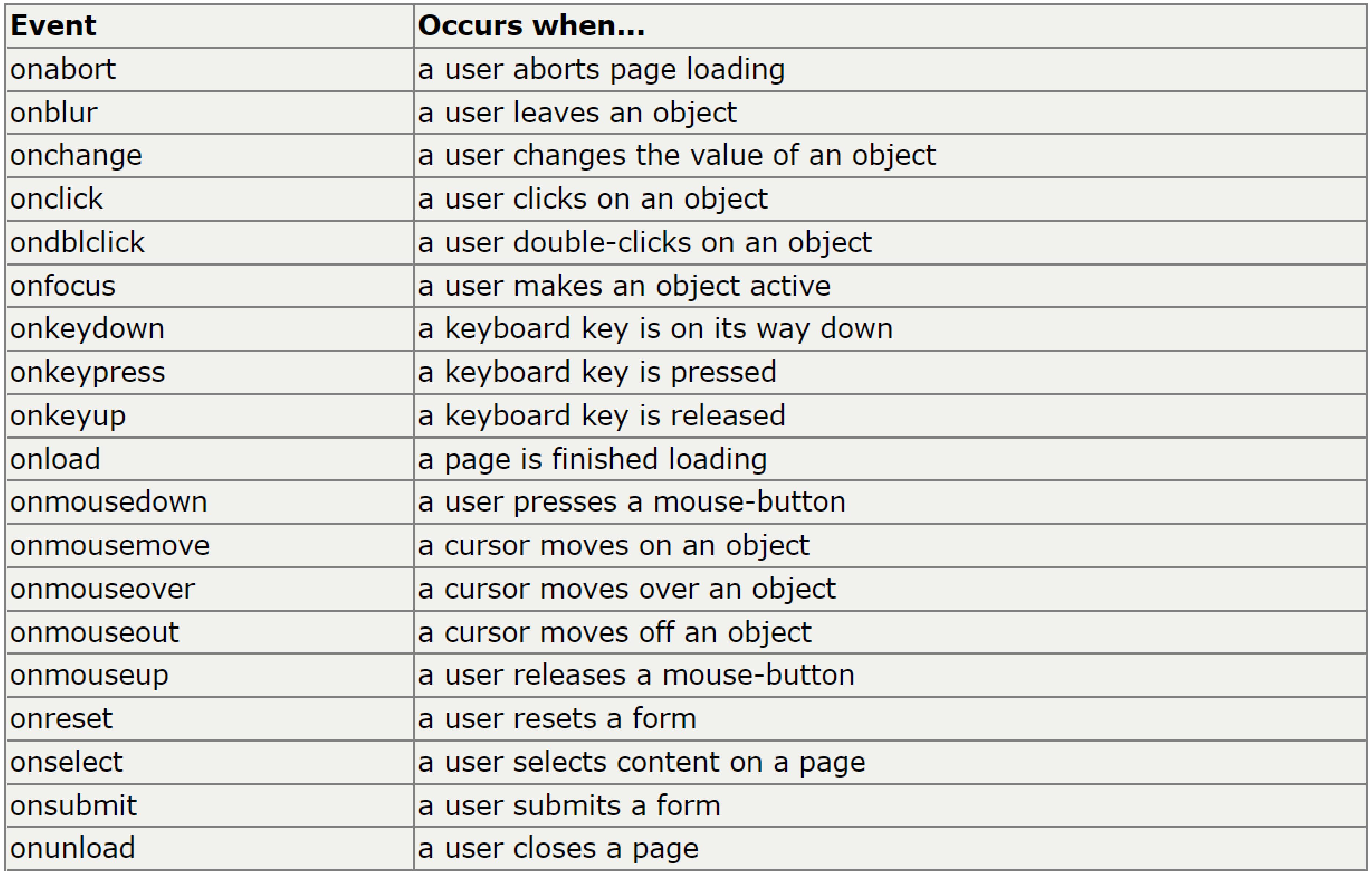
```
clicked
>
```

DOM communicates to JavaScript with Events

Event types:

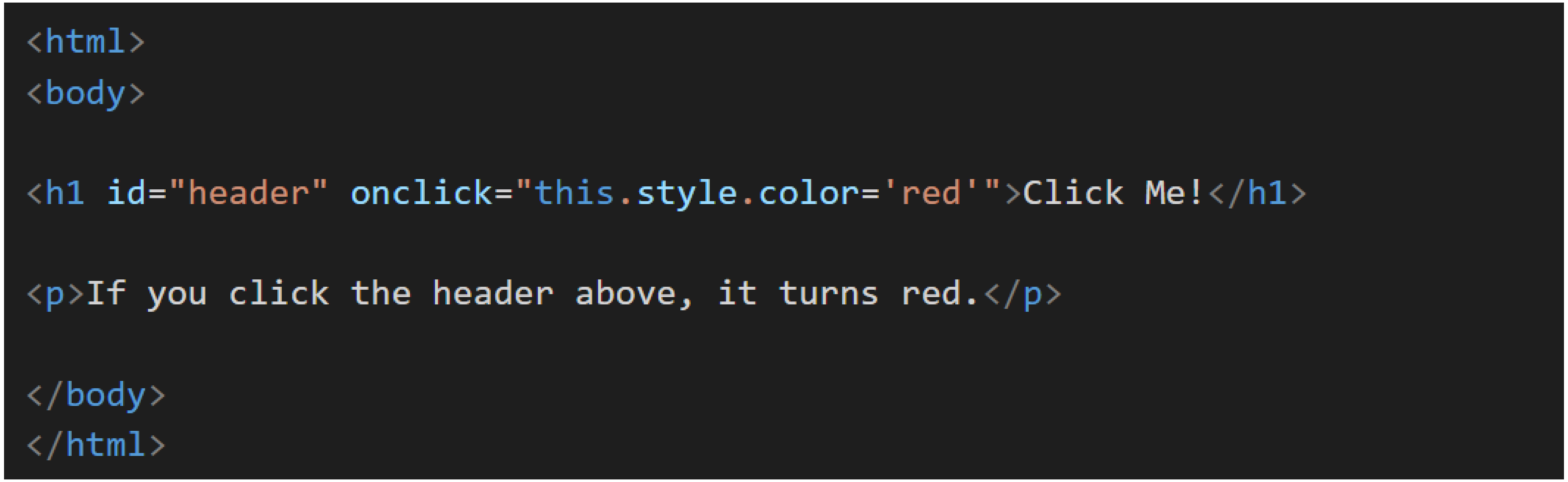
- Mouse-related: mouse movement, button click, enter/leave element
- Keyboard-related: down, up, press
- Focus-related: focus in, focus out (blur)
- Input field changed, Form submitted
- Timer events
- Miscellaneous:
    - Content of an element has changed
    - Page loaded/unloaded
    - Image loaded
    - Uncaught exception

List of Events

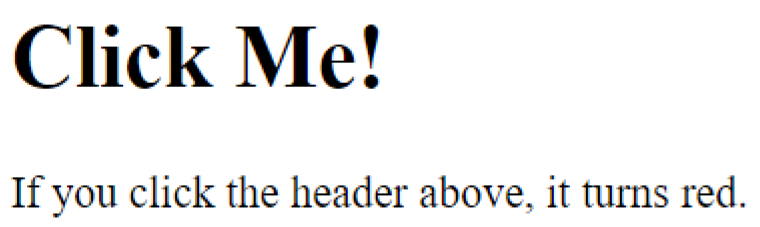


Event	Occurs when...
onabort	a user aborts page loading
onblur	a user leaves an object
onchange	a user changes the value of an object
onclick	a user clicks on an object
ondblclick	a user double-clicks on an object
onfocus	a user makes an object active
onkeydown	a keyboard key is on its way down
onkeypress	a keyboard key is pressed
onkeyup	a keyboard key is released
onload	a page is finished loading
onmousedown	a user presses a mouse-button
onmousemove	a cursor moves on an object
onmouseover	a cursor moves over an object
onmouseout	a cursor moves off an object
onmouseup	a user releases a mouse-button
onreset	a user resets a form
onselect	a user selects content on a page
onsubmit	a user submits a form
onunload	a user closes a page

DHTML CSS

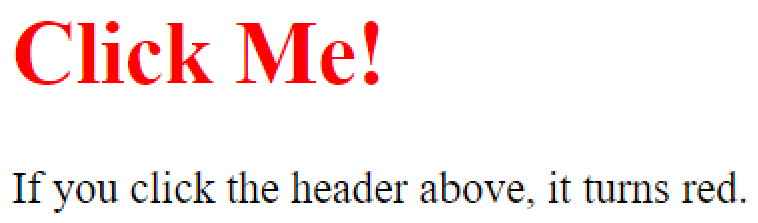


```html
<html>
<body>

<h1 id="header" onclick="this.style.color='red'">Click Me!</h1>

<p>If you click the header above, it turns red.</p>

</body>
</html>
```



Click Me!

If you click the header above, it turns red.



Click Me!

If you click the header above, it turns red.

Log messages aren't so interesting...

How do we interact with the page?

mer-defined function `square` (Part 1 of 2).

```
<?xml version = "1.0" encoding = "utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!-- Fig. 9.2: SquareInt.html -->
<!-- Programmer-defined function square. -->
<html xmlns = "http://www.w3.org/1999/xhtml">
   <head>
      <title>A Programmer-Defined square Function</title>
      <script type = "text/javascript">
         <!--
         document.writeln( "<h1>Square the numbers from 1 to 10</h1>" );

         // square the numbers from 1 to 10
         for ( var x = 1; x <= 10; x++ )
            document.writeln( "The square of " + x + " is " +
               square( x ) + "<br />" );

         // The following square function definition is executed
         // only when the function is explicitly called.

         // square function definition
         function square( y )
         {
            return y * y;
         } // end function square
         // -->
      </script>
   </head><body></bod
</html>
```

Calls function `square` with `x` as an argument, which will return the value to be inserted here

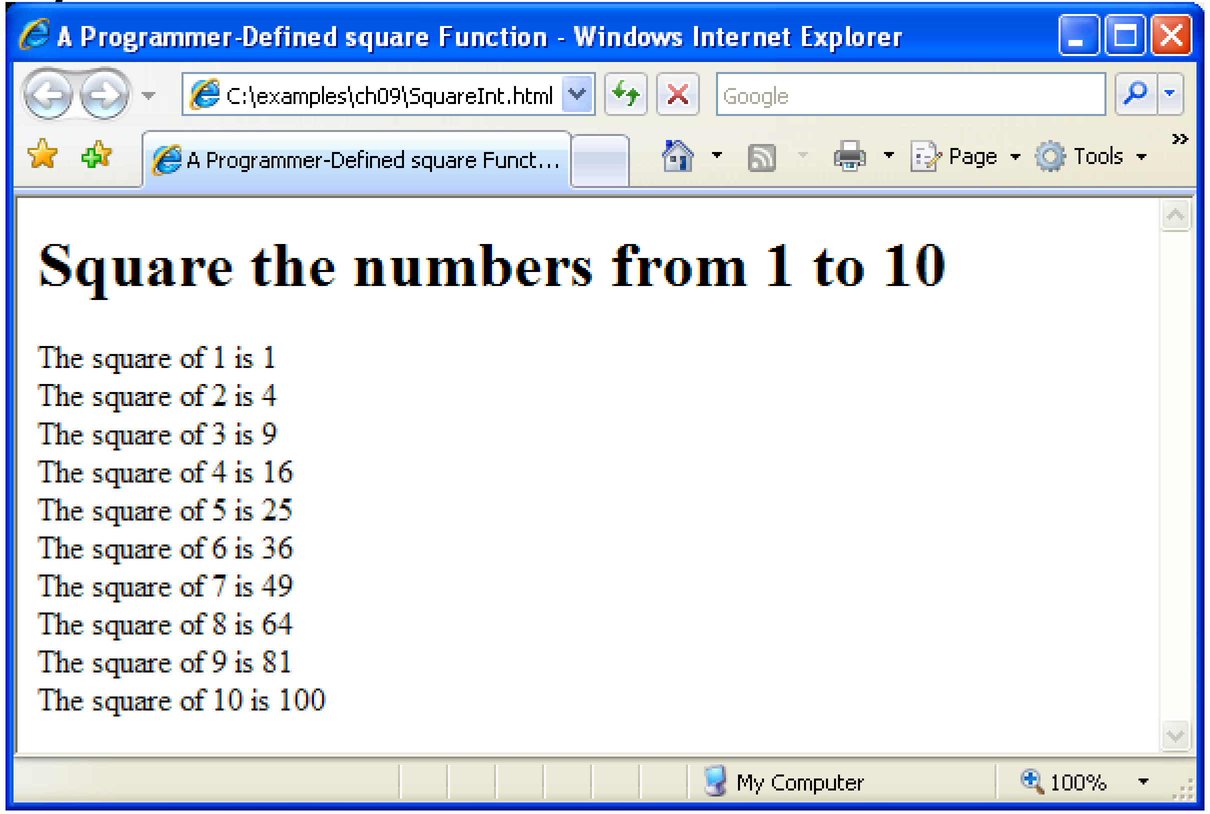
Begin function `square`

Names the parameter `y`

Returns the value of `y * y` (the argument squared) to the caller

End function `square`

Fig. 9.2 | Programmer-defined function `square` (Part 2 of 2).


A Programmer-Defined square Function - Windows Internet Explorer
C:\examples\ch09\SquareInt.html
Google
A Programmer-Defined square Funct...
Page
Tools
My Computer
100%

```
Square the numbers from 1 to 10

The square of 1 is 1
The square of 2 is 4
The square of 3 is 9
The square of 4 is 16
The square of 5 is 25
The square of 6 is 36
The square of 7 is 49
The square of 8 is 64
The square of 9 is 81
The square of 10 is 100
```

```
<?xml version = "1.0" encoding = "utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!-- Fig. 9.3: maximum.html -->
<!-- Programmer-Defined maximum function. -->
<html xmlns = "http://www.w3.org/1999/xhtml">
   <head>
      <title>Finding the Maximum of Three Values</title>
      <script type = "text/javascript">
         <!--
         var input1 = window.prompt( "Enter first number", "0" );
         var input2 = window.prompt( "Enter second number", "0" );
         var input3 = window.prompt( "Enter third number", "0" );

         var value1 = parseFloat( input1 );
         var value2 = parseFloat( input2 );
         var value3 = parseFloat( input3 );

```

Creates integer values from user input

mer-
defined
`maximum`
function
(Part 1 of 3).

mer-defined maximum function (Part 2

```
20             var maxValue = maximum( value1, value2, value3 );
21
22                         nt.writeln( "First number: " + value1 +
23                         r />Second number: " + value2 +
24                         r />Third number: " + value3 +
25                         r />Maximum is: " + maxValue );
26
27                         imum function definition (called from line 20)
28                         on maximum( x, y, z )
29             {
30                return Math.max( x, Math.max( y, z ) );
31             } // end function maximum
32             // -->
33          </script>
34       </head>
35       <body>
36          <p>Click Refresh (or Reload) to run the script again</p>
37       </body>
38    </html>
```

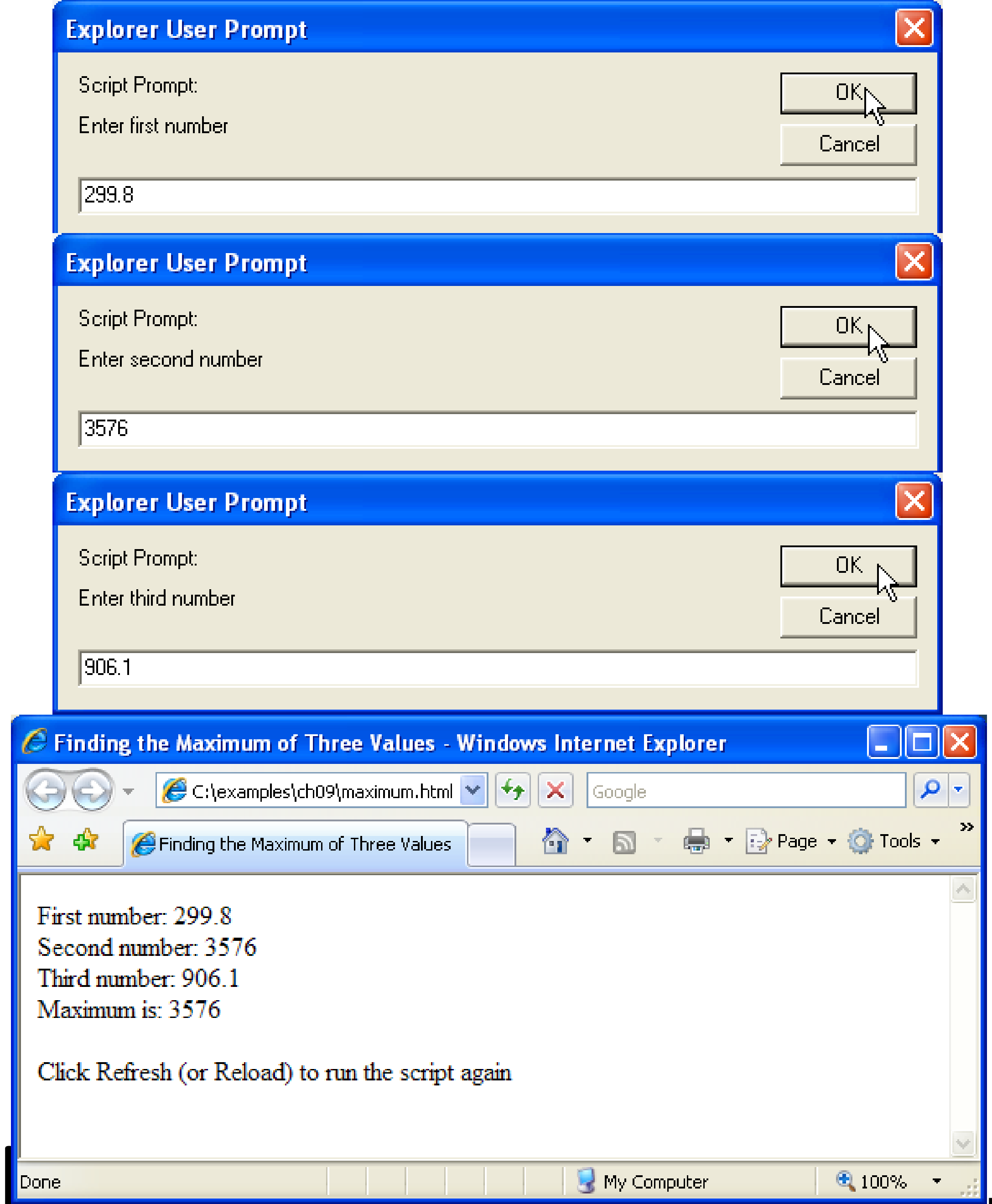
Variable `maxValue` stores the return value of the call to `maximum`

Calls function `maximum` with arguments `value1`, `value2` and `value3`

Begin function `maximum` with local variables `x`, `y` and `z`

Calls the `Math` object's method `max` to compare the first variable with the maximum of the other two

End function `maximum`


Explorer User Prompt
OK
Cancel
Finding the Maximum of Three Values - Windows Internet Explorer
C:\examples\ch09\maximum.html
Google
Finding the Maximum of Three Values
Page
Tools
Done
My Computer
100%

```
Script Prompt:
Enter first number
299.8

Script Prompt:
Enter second number
3576

Script Prompt:
Enter third number
906.1
```

```
First number: 299.8
Second number: 3576
Third number: 906.1
Maximum is: 3576

Click Refresh (or Reload) to run the script again
```

Fig. 9.3 | Programmer-defined `maximum` function (Part 3 of 3).

integers, shifting and scaling (Part 1 of 2).

```
<?xml version = "1.0" encoding = "utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!-- Fig. 9.4: RandomInt.html -->
<!-- Random integers, shifting and scaling. -->
<html xmlns = "http://www.w3.org/1999/xhtml">
   <head>
      <title>Shifted and Scaled Random Integers</title>
      <style type = "text/css">
         table { width: 50%;
                 border: 1px solid gray;
                 text-align: center }
      </style>
      <script type = "text/javascript">
         <!--
         var value;

         document.writeln( "<table>" );
         document.writeln( "<caption>Random Numbers</caption><tr>" );

         for ( var i = 1; i <= 20; i++ )
         {
            value = Math.floor( 1 + Math.random() * 6 );
            document.writeln( "<td>" + value + "</td>" );

            // start a new table row every 5 entries
            if ( i % 5 == 0 && i != 20 )
               document.writeln( "</tr><tr>" );
         } // end for

```

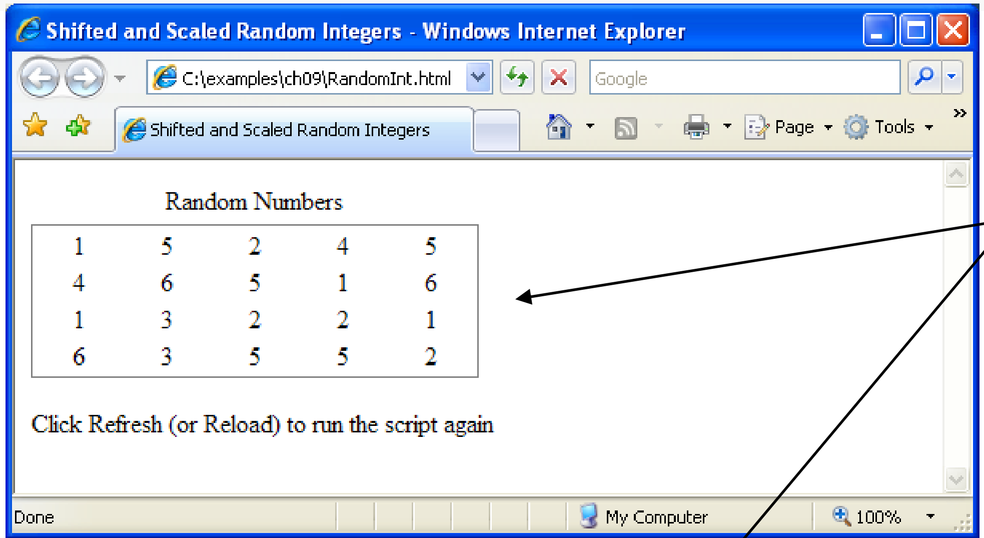
Shifts the range of return values up by 1

Scales the range of return values by a factor of 6

Takes the floor of the number from 1.0 up to, but not including, 7.0

integers, shifting and scaling (Part 2

```
         document.writeln( "</tr></table>" );
         // -->
      </script>
   </head>
   <body>
      <p>Click Refresh (or Reload) to run the script again</p>
   </body>
</html>
```



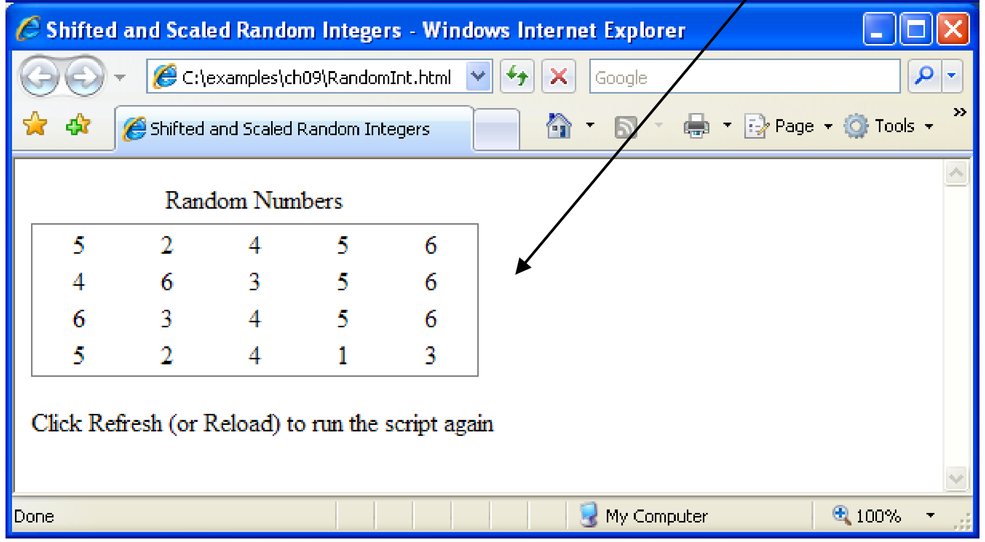
Shifted and Scaled Random Integers - Windows Internet Explorer

C:\examples\ch09\RandomInt.html

Random Numbers

1	5	2	4	5
4	6	5	1	6
1	3	2	2	1
6	3	5	5	2

Click Refresh (or Reload) to run the script again



Shifted and Scaled Random Integers - Windows Internet Explorer

C:\examples\ch09\RandomInt.html

Random Numbers

5	2	4	5	6
4	6	3	5	6
6	3	4	5	6
5	2	4	1	3

Click Refresh (or Reload) to run the script again

Variable `value` has a value from 1 to 6

JavaScript global functions.

Global function	Description
`escape`	Takes a string argument and returns a string in which all spaces, punctuation, accent characters and any other character that is not in the ASCII character set (see Appendix D, ASCII Character Set) are encoded in a hexadecimal format (see Appendix E, Number Systems) that can be represented on all platforms.
`eval`	Takes a string argument representing JavaScript code to execute. The JavaScript interpreter evaluates the code and executes it when the `eval` function is called. This function allows JavaScript code to be stored as strings and executed dynamically. [*Note:* It is considered a serious security risk to use eval to process any data entered by a user because a malicious user could exploit this to run dangerous code.]
`isFinite`	Takes a numeric argument and returns `true` if the value of the argument is not `NaN`, `Number.POSITIVE_INFINITY` or `Number.NEGATIVE_INFINITY` (values that are not numbers or numbers outside the range that JavaScript supports)—otherwise, the function returns `false`.
`isNaN`	Takes a numeric argument and returns `true` if the value of the argument is not a number; otherwise, it returns `false`. The function is commonly used with the return value of `parseInt` or `parseFloat` to determine whether the result is a proper numeric value.
`parseFloat`	Takes a string argument and attempts to convert the beginning of the string into a floating-point value. If the conversion is unsuccessful, the function returns NaN; otherwise, it returns the converted value (e.g., `parseFloat( "abc123.45" )` returns `NaN`, and `parseFloat( "123.45abc" )` returns the value `123.45`).
`parseInt`	Takes a string argument and attempts to convert the beginning of the string into an integer value. If the conversion is unsuccessful, the function returns NaN; otherwise, it returns the converted value (e.g., parseInt("abc123") returns NaN, and parseInt("123abc") returns the integer value 123). This function takes an optional second argument, from 2 to 36, specifying the **radix** (or **base**) of the number. Base 2 indicates that the first argument string is in **binary** format, base 8 indicates that the first argument string is in **octal** format and base 16 indicates that the first argument string is in **hexadecimal** format. See Appendix E, Number Systems, for more information on binary, octal and hexadecimal numbers.
`unescape`	Takes a string as its argument and returns a string in which all characters previously encoded with `escape` are decoded.