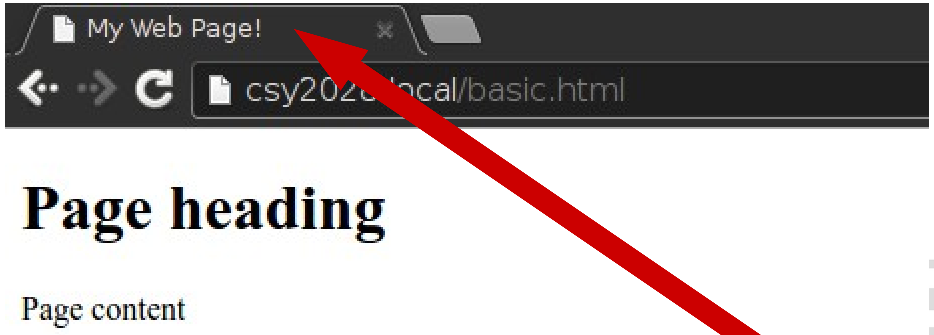
CSYM019
Internet Programming

Dr Muawya Eldaw
muawya.eldaw@northampton.ac.uk

HTML - Recap

- A basic HTML file looks like this:

```
<!DOCTYPE html>
<html>
  <head>
    <title>My Web Page!</title>
  </head>

  <body>
    <h1>Page heading</h1>
    <p>Page content</p>
  </body>
</html>
```



Page heading

Page content

Nothing in <head> appears on the page, but the info is used elsewhere

Javascript

- HTML and CSS can be used to display information and present it in a specific way
- However, CSS and HTML have limitations: You cannot change the contents of the page after it has been drawn on the screen
- HTML is not very interactive, you can display a page but not control what happens when the user interacts with it

Javascript

- HTML is a *Markup Language* this means it describes how data is *structured.*
- When the HTML code is run, it is interpreted by the browser to generate a visual output and run in order, line by line top to bottom
- Javascript is a *programming language.* This means you as the developer has control over how the program is executed, it’s not usually executed in *linear* fashion

Javascript

- Like CSS, Javascript code should be placed in its own file
- Javascript files have a .js extension
- To run javascript code (the code in .js file), the HTML page must reference the javascript file using a <script> HTML tag

Javascript

- The script tag has an src attribute (like the <img> tag) that points to the javascript file:
- <script> tags should be placed inside the page's <head> tag:

```
<!DOCTYPE html>
<html>
  <head>
    <title>My Web Page!</title>
    <script src="script.js"></script>
  </head>

  <body>
    <h1>Page heading</h1>
    <p>Page content</p>
  </body>
</html>
```

There has been some debate about whether to place the <script> tag in the head or body tag.

For modern browsers, the <head> tag is preferred.

Look up HTTP2.0 push if you are interested

<script> tag

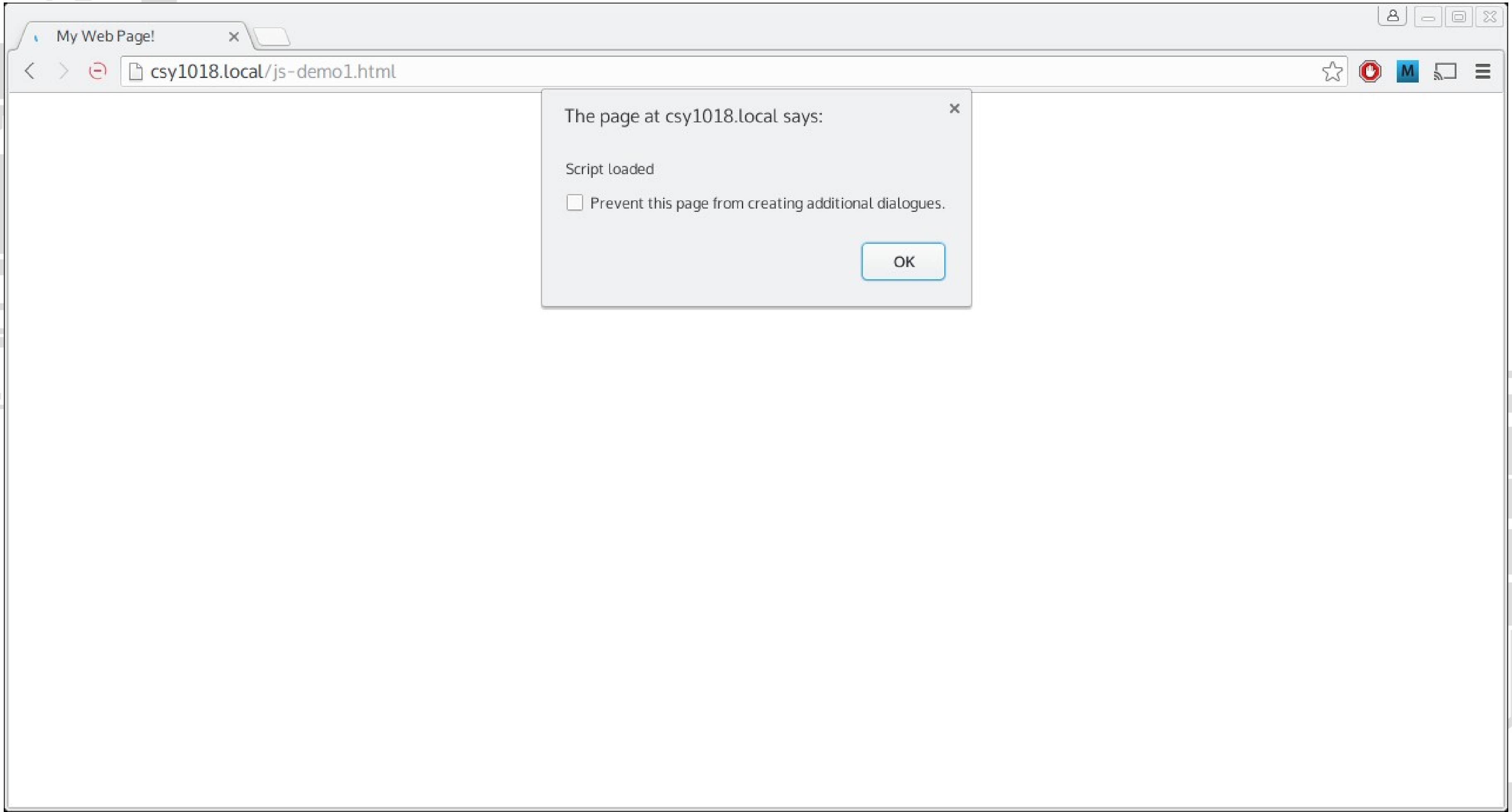
- The script tag does not need anything between <script> and </script> but both start and end tags are required
- <script src=”file.js” /> **will not work in all browsers, you must use <script src=”file.js”></script>**

Javascript

- To check your script.js is loading correctly you can add some code to its
- Javascript includes the function *alert*
- The *alert* function lets you create a pop up alert box to display some text
- Using this code in *script.js:*

```
alert('Script loaded');
```

Script.js


My Web Page!

csy1018.local/js-demo1.html

The page at csy1018.local says:

Script loaded

Prevent this page from creating additional dialogues.

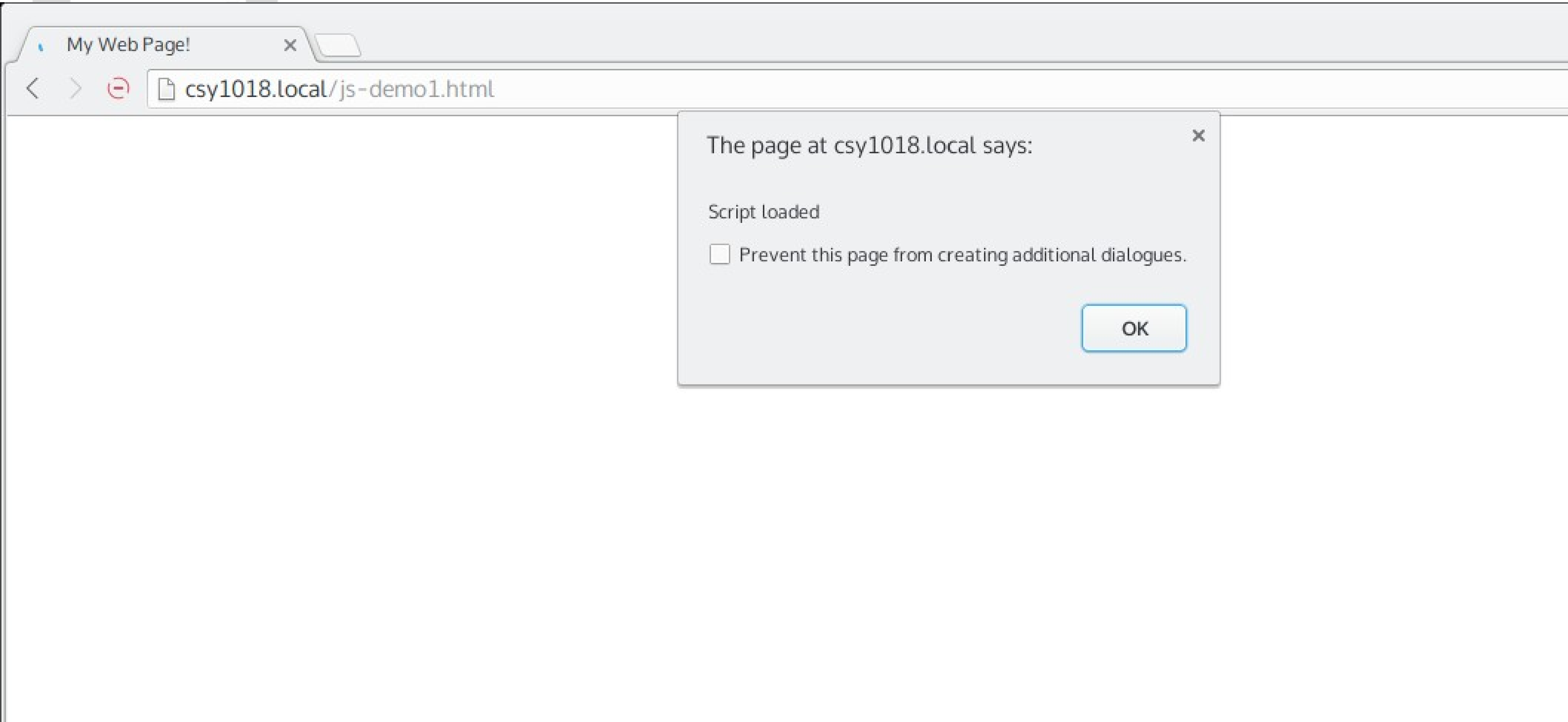
Alert

- This will look different in each browser as the browser controls what happens when alert() is called

Javascript

- You'll notice that when the alert popup appears that the contents of the page are not visible behind it
- Once you click 'OK' the contents of the page appear
- This is because the Javscript runs before the page has been drawn on the screen

Script.js


My Web Page!
csy1018.local/js-demo1.html

The page at csy1018.local says:

Script loaded

Prevent this page from creating additional dialogues.
OK

Javascript

- Javascript can be used to control HTML elements on the page
- Javascript can be used to:
  - Assign CSS to the element
  - Add or remove HTML attributes
  - Read the contents of form elements
  - Detect when an element is interacted with (moused over, clicked, typed into, etc)

Javascript - Variables

- Javascript allows you to give values labels
- A labelled value storage is called a variable and can store a single value
- To declare a variable use the code

```
var variableName = variableValue;
```

- You can give the variable any name you like, this is chosen by you, not javascript
- The value of the variable is also chosen by you
- The only parts that are defined by the language are the:
  - var keyword – this tells javascript you are creating a variable
  - = sign – this tells javascript you are writing a value to the variable

Javascript - Variables

- There are two main types of variable:
  - Numbers
  - Strings (text)
- To assign a number variable you can use

```
var numberVariable1 = 123;
var numberVariable2 = 123.45;
```

- To assign a string variable you must surround the string with quotes:

```
var stringVariable = 'Script loaded';
```

Javascript variables

- The code can be translated into english

```
var numberVariable1 = 123;
```

- *Var* can be read as “Create a variable called”
- And the = symbol can be read as “and set it to”
- The code above can be read as:
  - Create a variable called numberVariable1 and set it to 123

Variables

- Once you have stored a value inside a variable (labelled storage) you can reference it later, for example in the *alert* function

```
var stringVariable = 'Script loaded';
alert(stringVariable);
```

- This will create an alert box with the text “Script loaded”

Strings

- A string is a piece of data that will be used in the program
- Anything in quotes is treated as a string
- Generally, anything that's not inside quotes is treated as a javascript command
- Javascript sees the opening quote as "Start of string" and the next quote as end of string

```
var stringVariable = 'Script loaded';
var stringVariable = 'Script loaded';
```

First quote designates the start of the string

Second quote designates the end of the string

- Anything in quotes (a string) is *data* that will be worked on
- Anything outside of quotes is treated as a *command to be followed*

```
var stringVariable = Script loaded;
```

This would tell Javascript to run the commands: Script and Loaded

These are not valid commands!

```
var stringVariable = 'Script loaded';
```

This would tell Javascript to store the text "script loaded"

Variables

- If you put quotes around something it is treated as a string, not a variable

- What will be the output of the following code?

```
var stringVariable = 'Script loaded';
alert('stringVariable');
```

Variables

- What will be the output of the following code?

```
var stringVariable = 'Script loaded';
alert('stringVariable');
```

- Because the variable name is in quotes, the variable name will be printed rather than its contents
- Instead the code should be

```
var stringVariable = 'Script loaded';
alert(stringVariable);
```

Commands vs data

- Remember:
  - Anything in quotes is *data* to be processed.
    - e.g. Text you wish to print on the screen
  - Anything not in quotes is a *command*
    - *E.g. the name of a “create a variable with this name”*
- It is important that you understand the difference as this is a very common problem beginners face

Javascript

- Note that each statement must be ended with a semicolon.
- The semicolon means “end of statement” and can be thought of like a full stop in an English sentence.

Other general notes about javascript

- Spelling matters! Even a slight typo will stop the entire script from running
- Javascript is *cAsE SenSItive.* If you create a variable with uppercase letters you must reference it with uppercase letters throughout your program!
- Missing a bracket, semicolon, dot, or other seemingly trivial mistake will prevent your code from working. Get used to looking for the tiniest of mistakes!

Variables

- You can create as many variables as you like
- Once a variable is created you can perform operations on it
- For numeric variables you can perform mathematical operations e.g.

```
var num1 = 5;
var num2 = 6;

var num3 = num1 + num2;

alert(num3);
```

Prints “11”

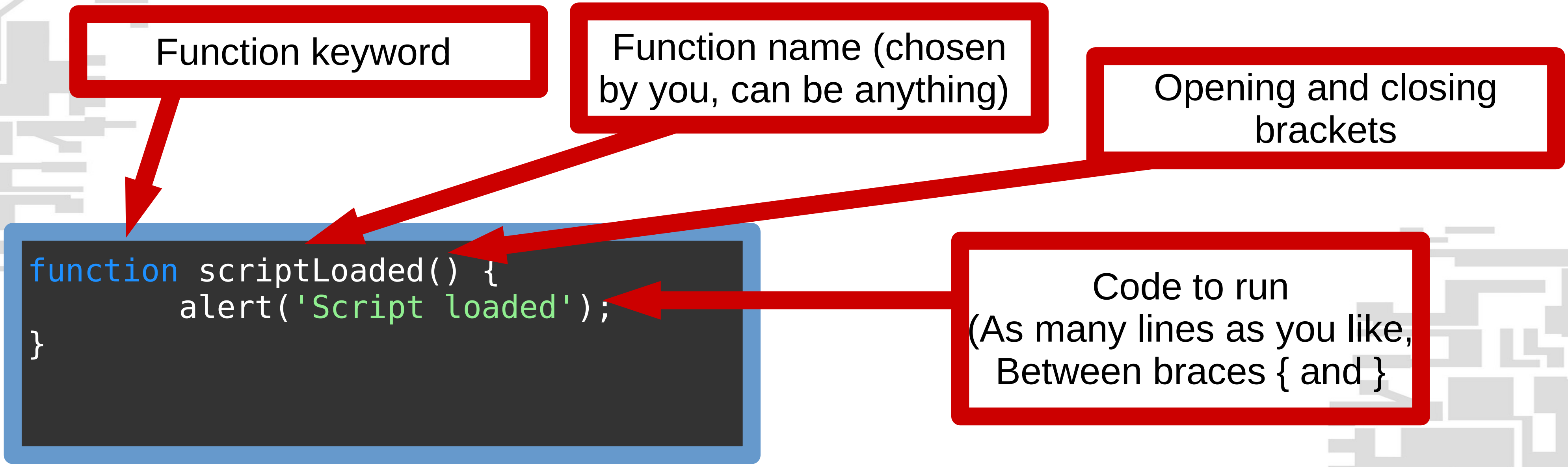
Functions

- You can *label* a block of code using a *function*
- This will store the code for later use where it can be referenced and run
- This allows you to write code out of sequence

```
function scriptLoaded() {
        alert('Script loaded');
}

function addition() {
        var num1 = 5;
        var num2 = 6;

        var num3 = num1 + num2;

        alert(num3);
}
```

Functions

- The syntax for a function looks like this:


Function keyword
Function name (chosen by you, can be anything)
Opening and closing brackets
Code to run (As many lines as you like, Between braces { and }

```
function scriptLoaded() {
        alert('Script loaded');
}
```

Functions

- When code is stored inside a function it is not executed as it's defined

```
function scriptLoaded() {
        alert('Script loaded');
}

function addition() {
        var num1 = 5;
        var num2 = 6;

        var num3 = num1 + num2;

        alert(num3);
}
```

- This will not display either alert box!

Functions

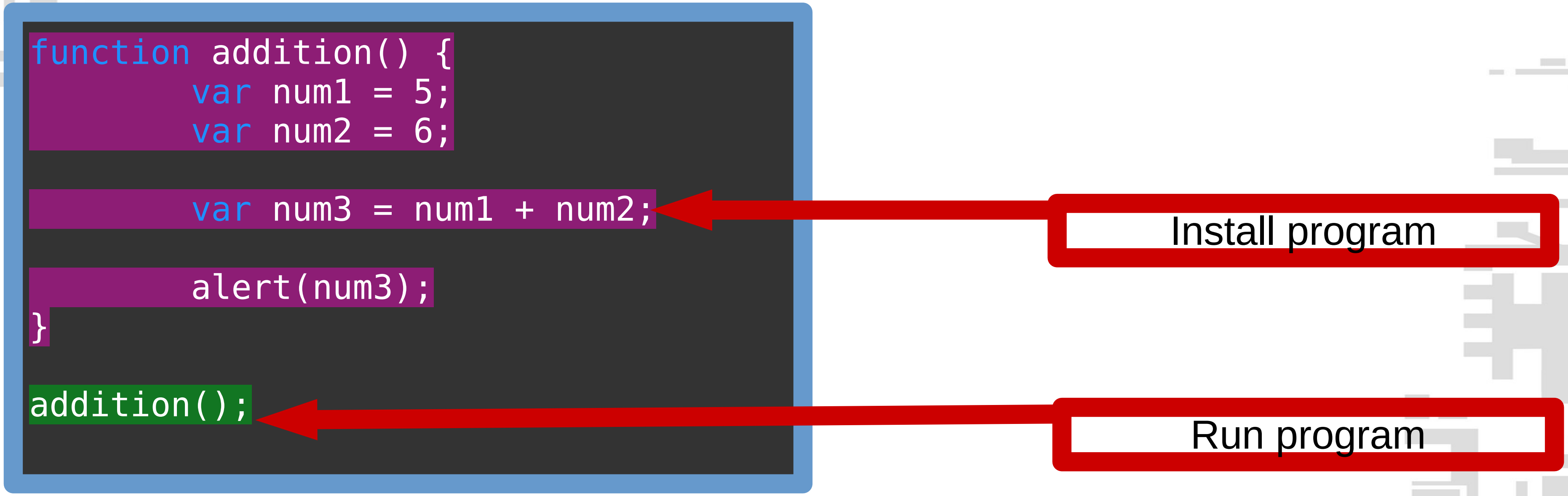
- Once a function has been defined it has to be *called*
- A function is called using the name followed by brackets
- Normally code gets run in the order it is written
- Functions allow you to run code in a different order

Functions

- To run the code in the addition function, it must be *called* using the code
-
```
addition();
```

Functions

- You can think of a function like a program on your computer
- Creating a function is like installing the program
- It is there on the computer but doesn’t do anything until you run it



```
function addition() {
        var num1 = 5;
        var num2 = 6;

        var num3 = num1 + num2;

        alert(num3);
}

addition();
```

Functions

- The javascript interpreter will ignore code inside function blocks until it is called:

```
function scriptLoaded() {
        alert('Script loaded');
}

function addition() {
        var num1 = 5;
        var num2 = 6;

        var num3 = num1 + num2;

        alert(num3);
}

addition();
```

Interpreter Position

Functions

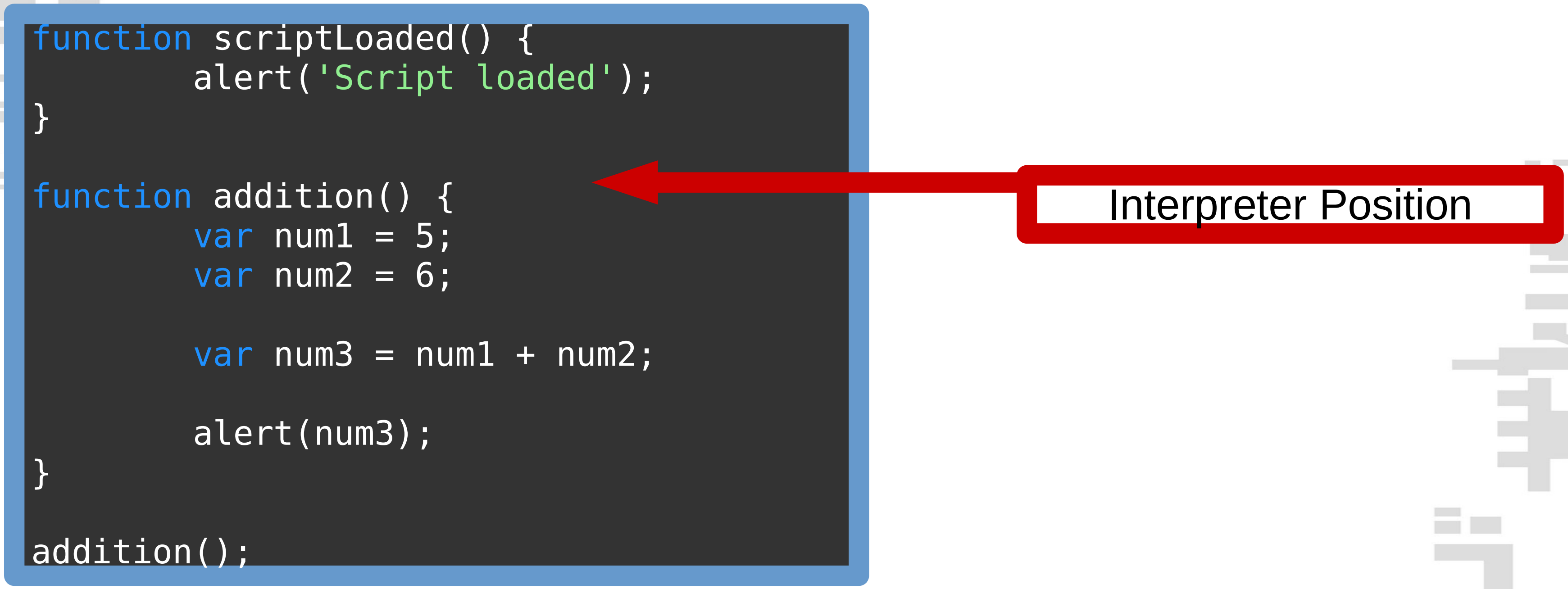
- The javascript interpreter will ignore code inside function blocks until it is called:

```
function scriptLoaded() {
        alert('Script loaded');
}

function addition() {
        var num1 = 5;
        var num2 = 6;

        var num3 = num1 + num2;

        alert(num3);
}

addition();
```

Interpreter Position

Functions

- The javascript interpreter will ignore code inside function blocks until it is called:



```
function scriptLoaded() {
        alert('Script loaded');
}

function addition() {
        var num1 = 5;
        var num2 = 6;

        var num3 = num1 + num2;

        alert(num3);
}

addition();
```

Functions

- The javascript interpreter will ignore code inside function blocks until it is called:

```
function scriptLoaded() {
        alert('Script loaded');
}

function addition() {
        var num1 = 5;
        var num2 = 6;

        var num3 = num1 + num2;

        alert(num3);
}

addition();
```

Interpreter Position

Functions

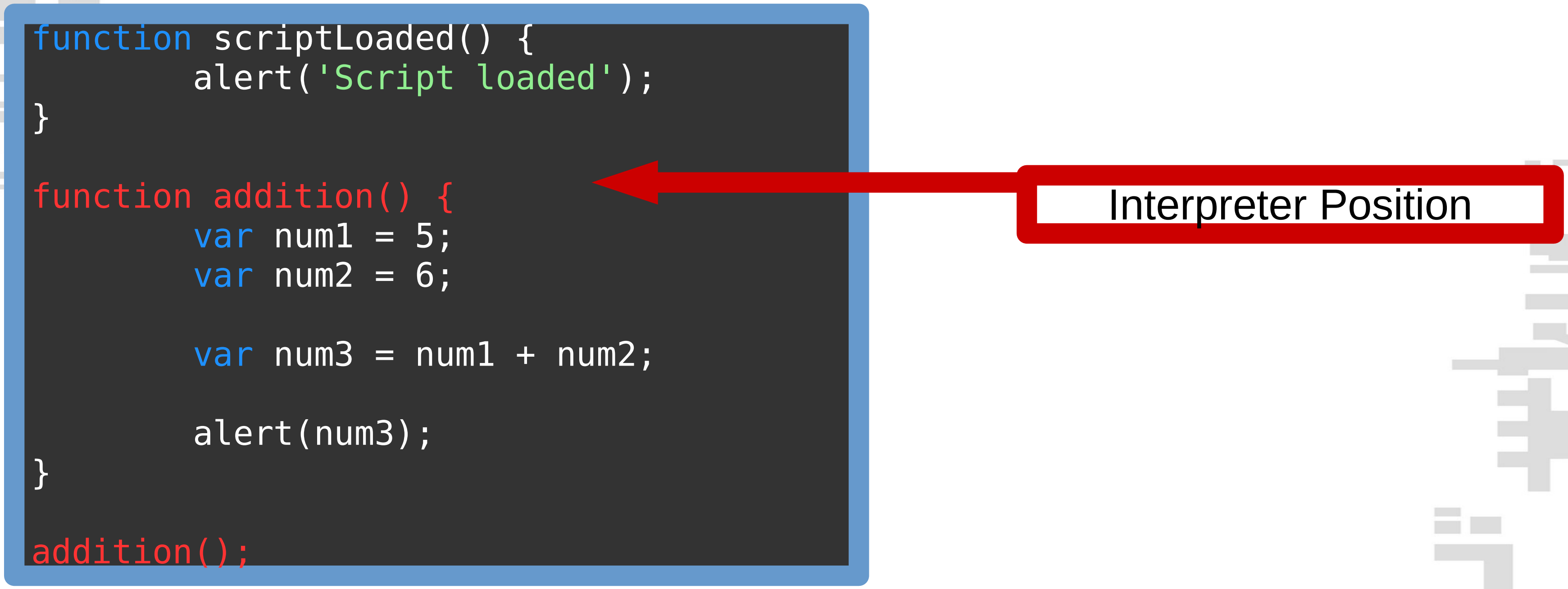
- The javascript interpreter will ignore code inside function blocks until it is called:

```
function scriptLoaded() {
        alert('Script loaded');
}

function addition() {
        var num1 = 5;
        var num2 = 6;

        var num3 = num1 + num2;

        alert(num3);
}

addition();
```

Interpreter Position

Functions

- The javascript interpreter will ignore code inside function blocks until it is called:



```
function scriptLoaded() {
        alert('Script loaded');
}

function addition() {
        var num1 = 5;
        var num2 = 6;

        var num3 = num1 + num2;

        alert(num3);
}

addition();
```

Functions

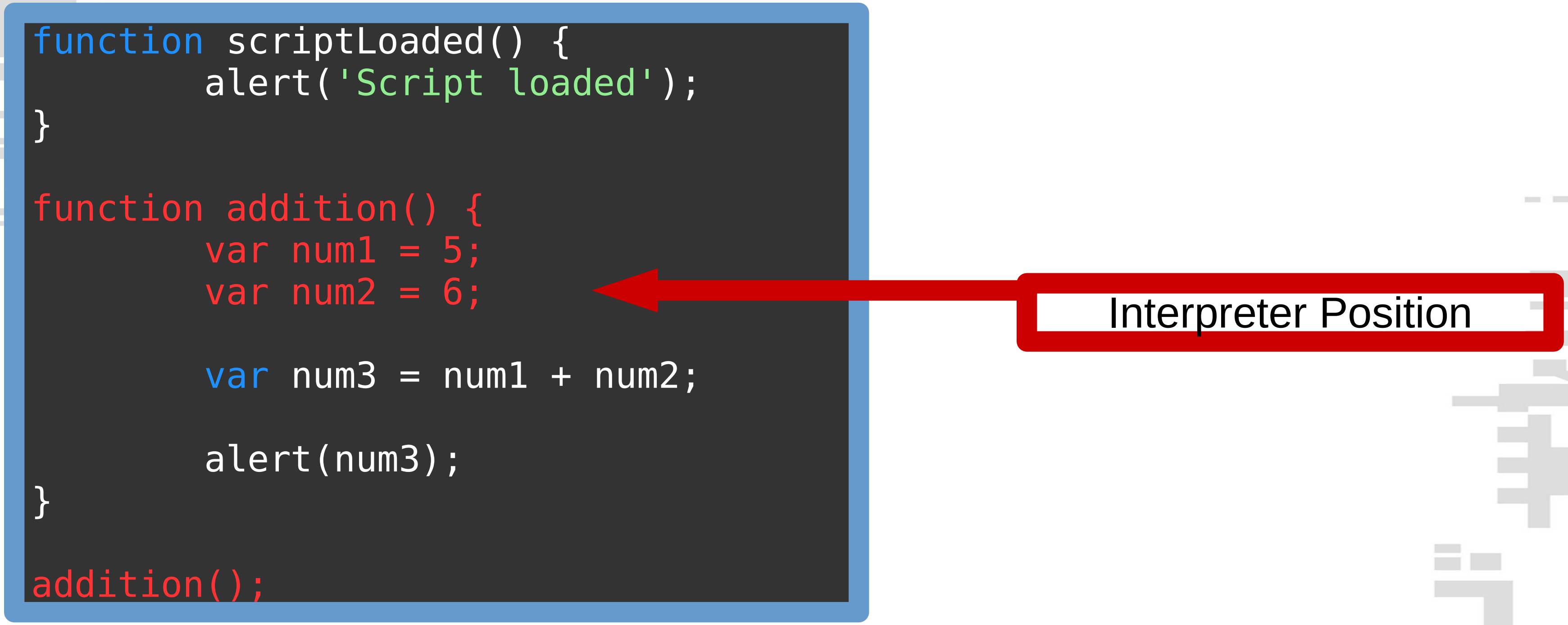
- The javascript interpreter will ignore code inside function blocks until it is called:

```
function scriptLoaded() {
        alert('Script loaded');
}

function addition() {
        var num1 = 5;
        var num2 = 6;

        var num3 = num1 + num2;

        alert(num3);
}

addition();
```

Interpreter Position

Functions

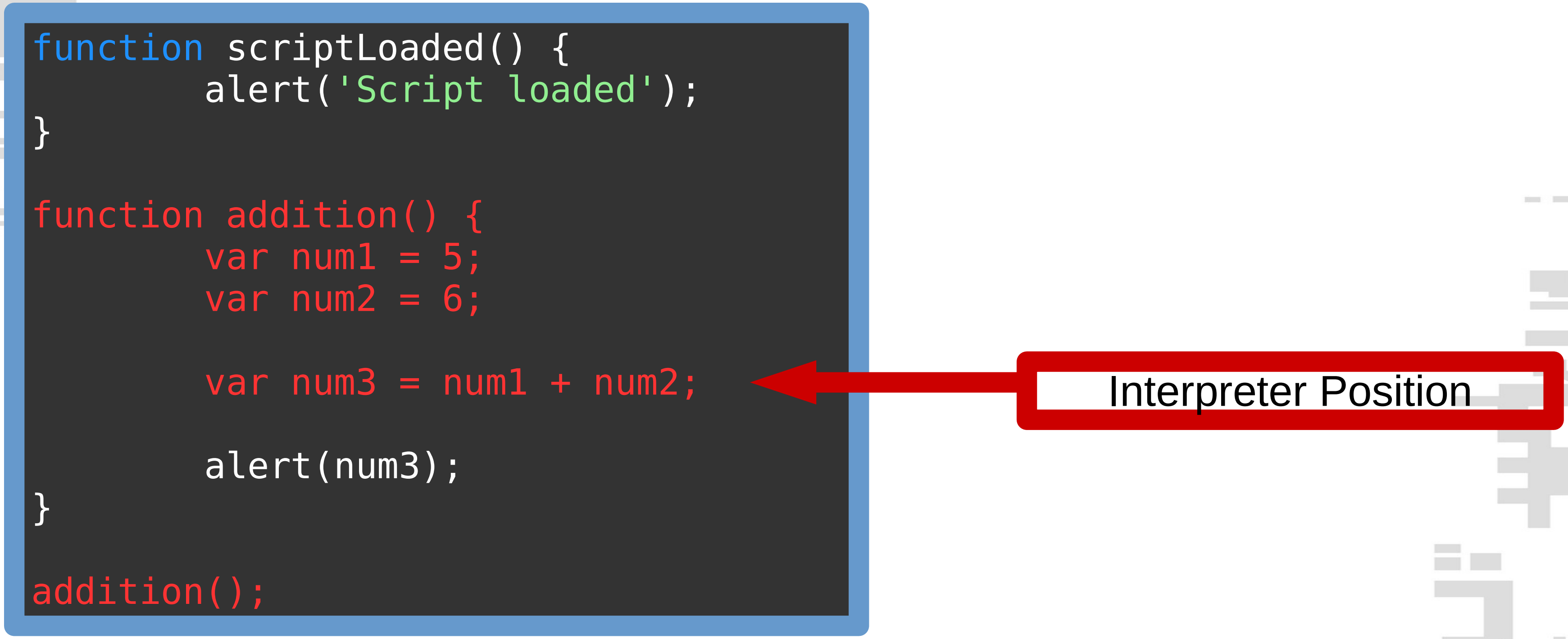
- The javascript interpreter will ignore code inside function blocks until it is called:



```javascript
function scriptLoaded() {
        alert('Script loaded');
}

function addition() {
        var num1 = 5;
        var num2 = 6;

        var num3 = num1 + num2;

        alert(num3);
}

addition();
```

Functions

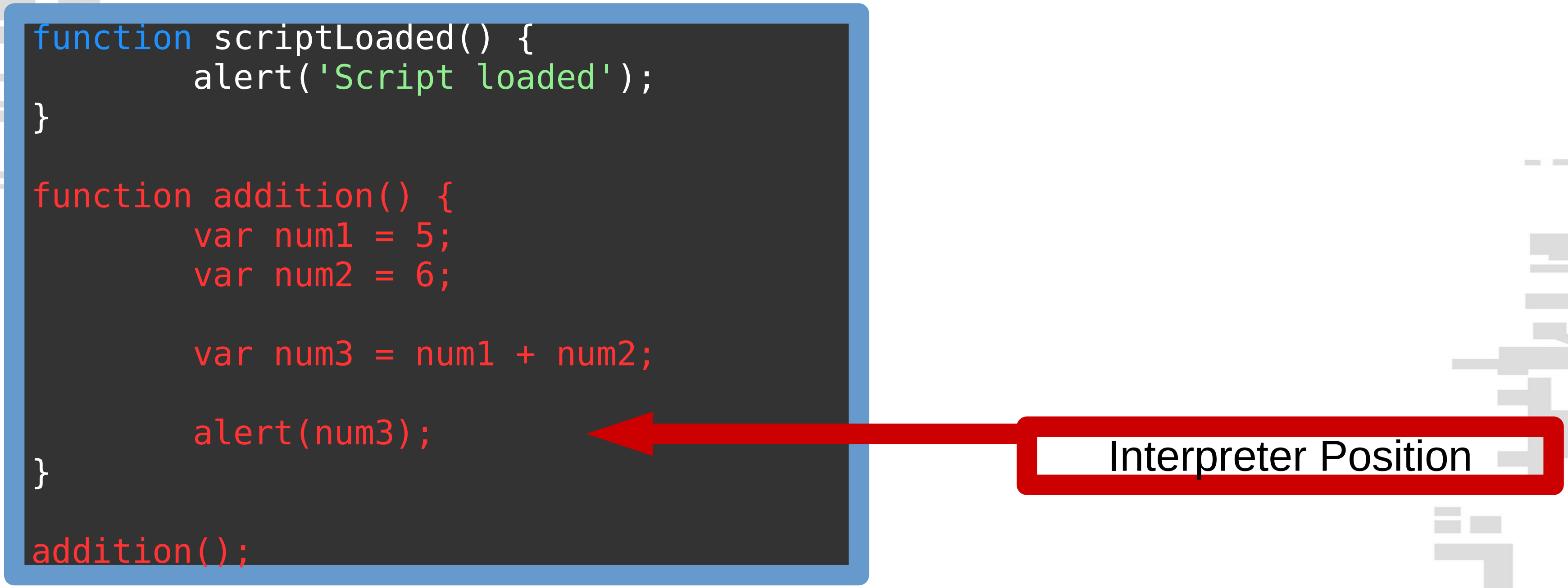
- The javascript interpreter will ignore code inside function blocks until it is called:



```
function scriptLoaded() {
        alert('Script loaded');
}

function addition() {
        var num1 = 5;
        var num2 = 6;

        var num3 = num1 + num2;

        alert(num3);
}

addition();
```

Functions

- The javascript interpreter will ignore code inside function blocks until it is called:



```
function scriptLoaded() {
        alert('Script loaded');
}

function addition() {
        var num1 = 5;
        var num2 = 6;

        var num3 = num1 + num2;

        alert(num3);
}

addition();
```

Functions

- If code is inside a function but the function is never *called* the code inside the function will never run

```
function scriptLoaded() {
        alert('Script loaded');
}

function addition() {
        var num1 = 5;
        var num2 = 6;

        var num3 = num1 + num2;

        alert(num3);
}

addition();
```

This code never gets executed

- You can think of functions like apps on your computer

```
function scriptLoaded() {
        alert('Script loaded');
}

function addition() {
        var num1 = 5;
        var num2 = 6;

        var num3 = num1 + num2;

        alert(num3);
}

addition();
```

There are two apps,
One titled “scriptLoaded”
One titled “addition”

This instruction tells
the browser to run
the app called “addition”

Functions

- This allows you to repeat code by calling the function more than once
- Putting repeated code inside a function is quicker and easier than typing it out twice but will produce the same result

Functions

- Both these scripts will produce the same output

```
function addition() {
        var num1 = 5;
        var num2 = 6;

        var num3 = num1 + num2;

        alert(num3);
}

addition();
addition();
```

```
        var num1 = 5;
        var num2 = 6;

        var num3 = num1 + num2;

        alert(num3);


        var num1 = 5;
        var num2 = 6;

        var num3 = num1 + num2;

        alert(num3);
```

Functions

```
function addition() {
        var num1 = 5;
        var num2 = 6;

        var num3 = num1 + num2;

        alert(num3);
}

addition();
addition();
```

Interpreter Position

Functions

```
function addition() {
        var num1 = 5;
        var num2 = 6;

        var num3 = num1 + num2;

        alert(num3);
}

addition();
addition();
```

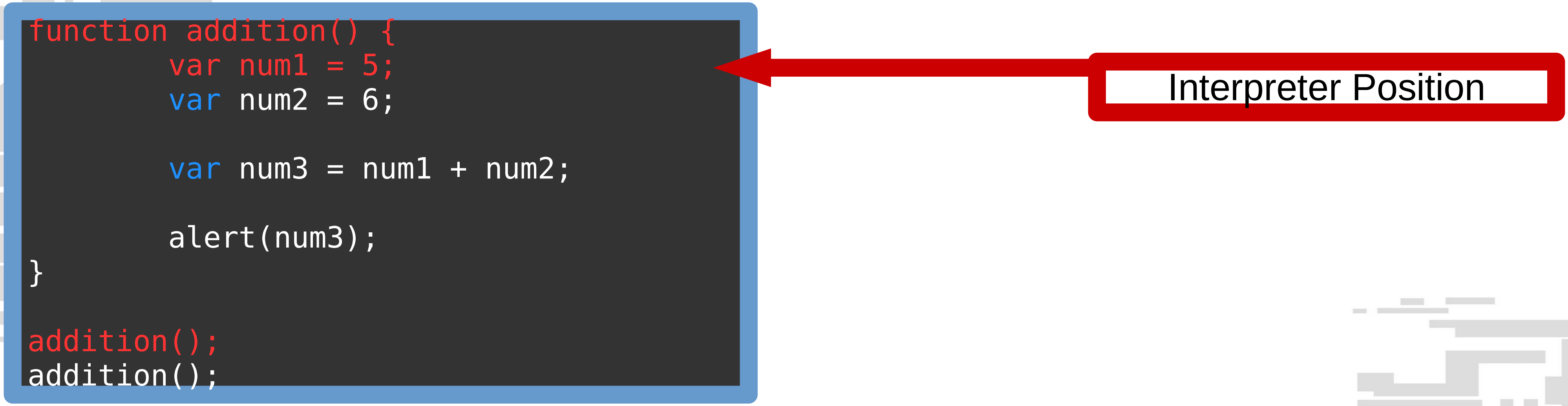
Interpreter Position

Functions

```
function addition() {
          var num1 = 5;
          var num2 = 6;

          var num3 = num1 + num2;

          alert(num3);
}

addition();
addition();
```

Interpreter Position

Functions



```
function addition() {
        var num1 = 5;
        var num2 = 6;

        var num3 = num1 + num2;

        alert(num3);
}

addition();
addition();
```

Functions

```
function addition() {
        var num1 = 5;
        var num2 = 6;

        var num3 = num1 + num2;

        alert(num3);
}

addition();
addition();
```

Interpreter Position

Functions

```
function addition() {
        var num1 = 5;
        var num2 = 6;

        var num3 = num1 + num2;

        alert(num3);
}

addition();
addition();
```

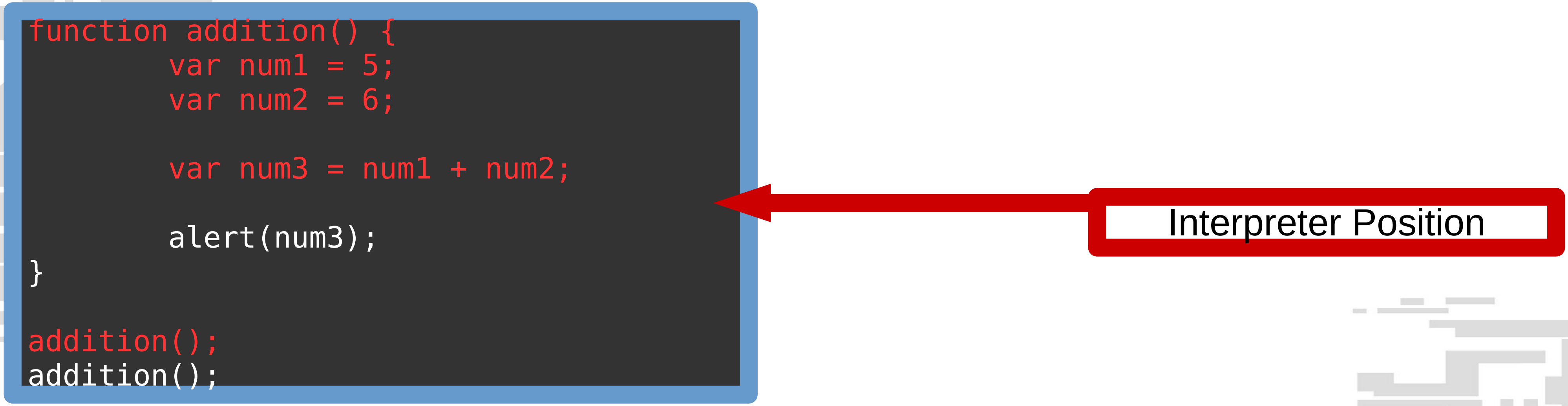
Interpreter Position

Functions

```
function addition() {
        var num1 = 5;
        var num2 = 6;

        var num3 = num1 + num2;

        alert(num3);
}

addition();
addition();
```

Interpreter Position

Functions



```
function addition() {
        var num1 = 5;
        var num2 = 6;

        var num3 = num1 + num2;

        alert(num3);
}

addition();
addition();
```

Functions

```
function addition() {
        var num1 = 5;
        var num2 = 6;

        var num3 = num1 + num2;

        alert(num3);
}

addition();
addition();
```

Interpreter Position

Functions

```
function addition() {
        var num1 = 5;
        var num2 = 6;

        var num3 = num1 + num2;

        alert(num3);
}

addition();
addition();
```

Interpreter Position

Functions

```
function addition() {
        var num1 = 5;
        var num2 = 6;

        var num3 = num1 + num2;

        alert(num3);
}

addition();
addition();
```

Interpreter Position

Functions

```
function addition() {
        var num1 = 5;
        var num2 = 6;

        var num3 = num1 + num2;

        alert(num3);
}

addition();
addition();
```

Interpreter Position

Functions

```
function addition() {
        var num1 = 5;
        var num2 = 6;

        var num3 = num1 + num2;

        alert(num3);
}

addition();
addition();
```

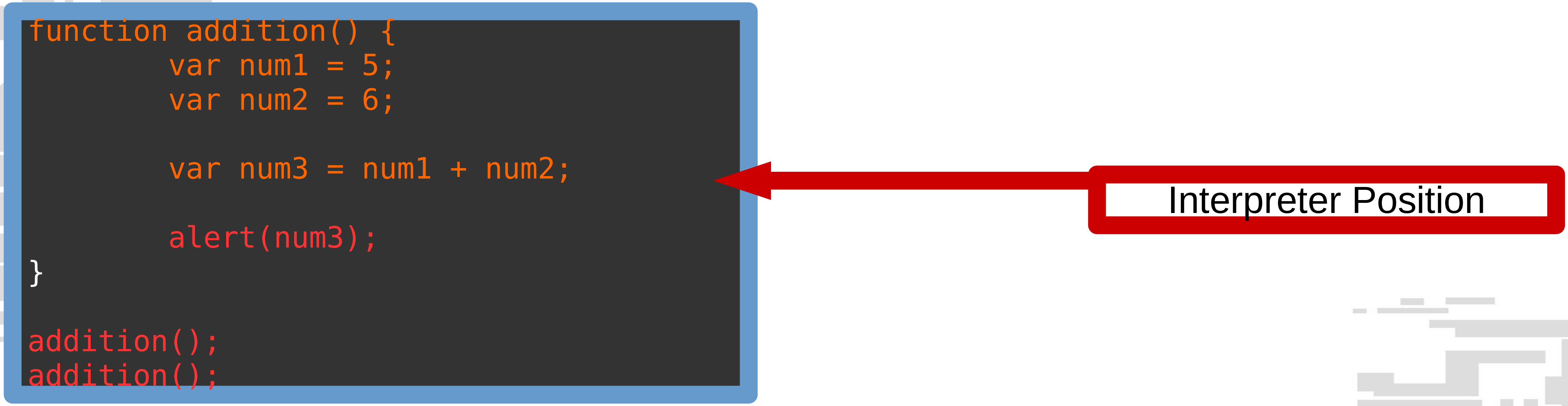
Interpreter Position

Functions

```
function addition() {
        var num1 = 5;
        var num2 = 6;

        var num3 = num1 + num2;

        alert(num3);
}

addition();
addition();
```

Interpreter Position

Functions



```
function addition() {
        var num1 = 5;
        var num2 = 6;

        var num3 = num1 + num2;

        alert(num3);
}

addition();
addition();
```

Functions

```
function addition() {
        var num1 = 5;
        var num2 = 6;

        var num3 = num1 + num2;

        alert(num3);
}

addition();
addition();
```

Interpreter Position

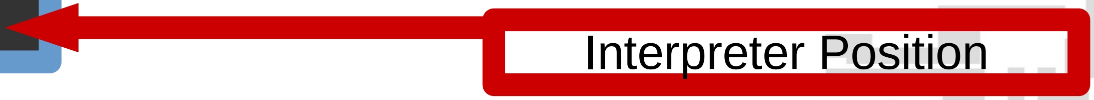
Functions

```
function addition() {
        var num1 = 5;
        var num2 = 6;

        var num3 = num1 + num2;

        alert(num3);
}

addition();
addition();
```

Interpreter Position

Functions

```
function addition() {
        var num1 = 5;
        var num2 = 6;

        var num3 = num1 + num2;

        alert(num3);
}

addition();
addition();
```



Exercise 1

- **5-10 mins**
- 1) Write a *function* called *print5* that has 5 alerts, the numbers 1 – 5 in their own alert box
- Call the function twice
- When the page loads you should get 10 alert popups in total.

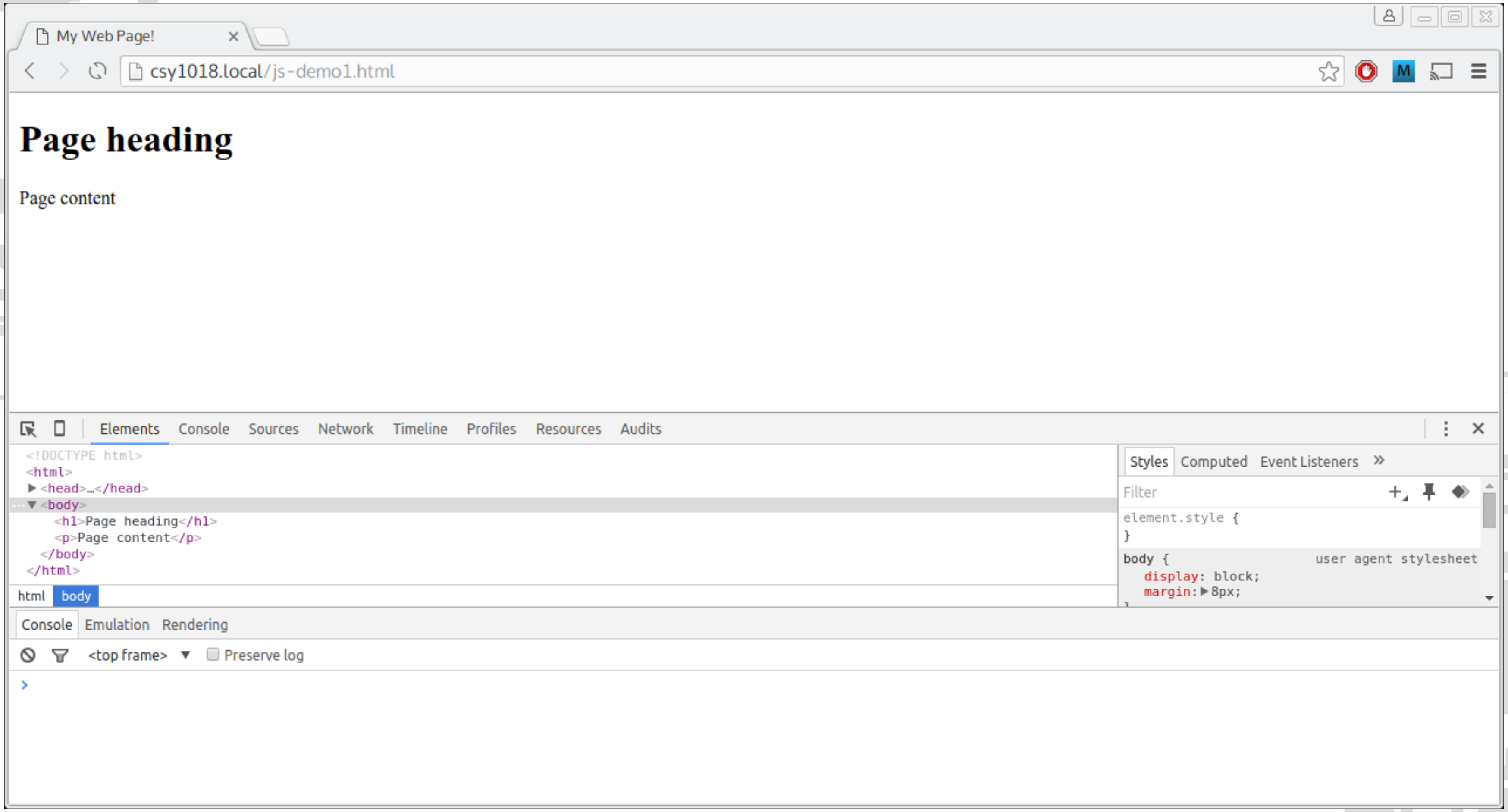
Exercise 1 - Solution

```
function print5() {
        alert(1);
        alert(2);
        alert(3);
        alert(4);
        alert(5);
}


print5();
print5();
```

Alert is annoying!

- Clicking through all those alerts can be annoying
- You can print to the *javascript console*
- This is a tool available in browser's developer tools
- In most browsers you can open it by pressing F12 on the keyboard
- This will open a panel in the browser (usually at the bottom) with a lot of options

Console



Page heading

Page content

```
<!DOCTYPE html>
<html>
▶<head>…</head>
▼<body>
    <h1>Page heading</h1>
    <p>Page content</p>
  </body>
</html>
```

```
element.style {
}
body {                    user agent stylesheet
  display: block;
  margin:▶8px;
```

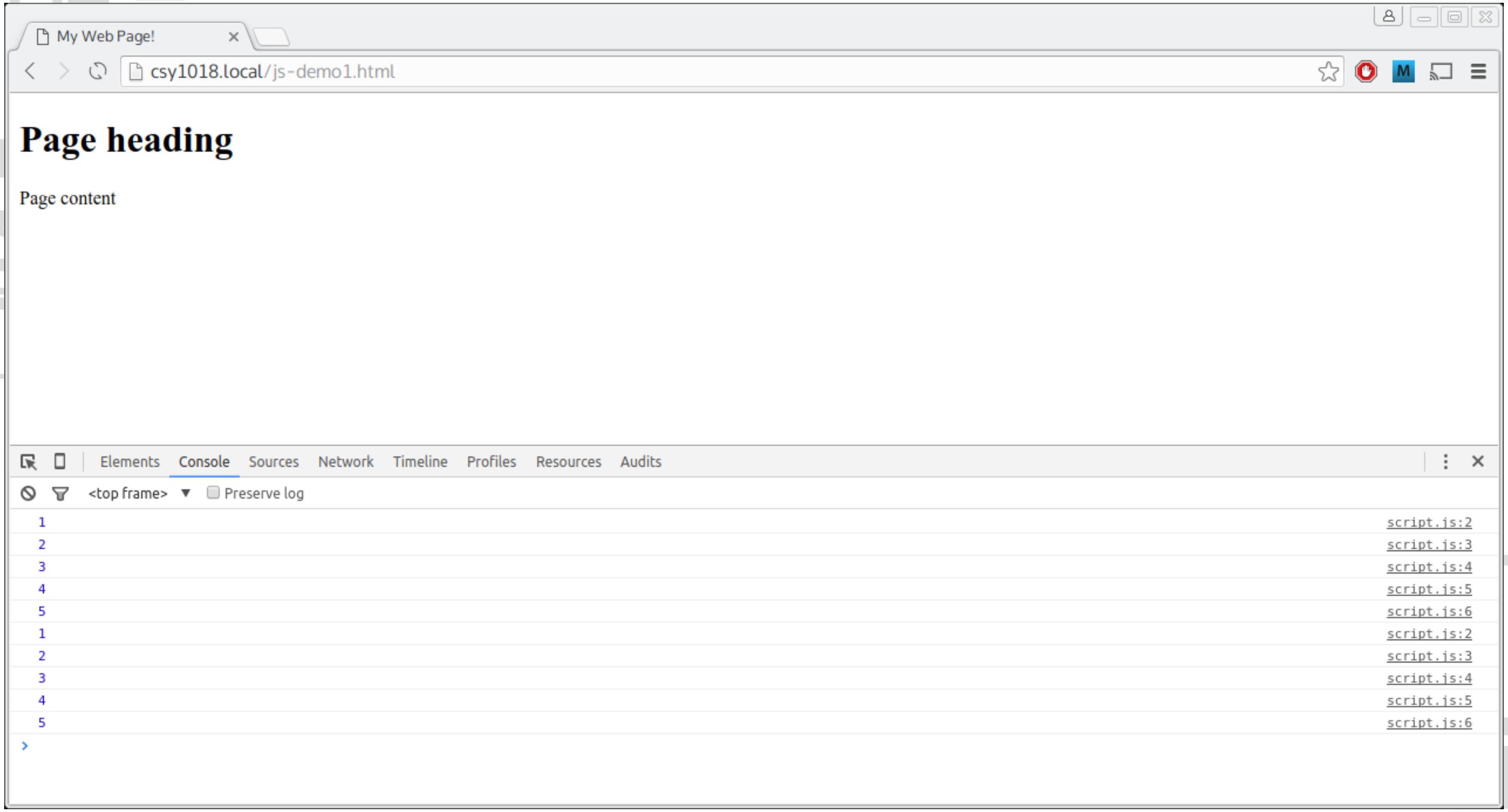
Console.log

- By changing alert() to console.log() you can write to the console instead of having to click ok on all the alerts!

```
function print5() {
        console.log(1);
        console.log(2);
        console.log(3);
        console.log(4);
        console.log(5);
}


print5();
print5();
```

Console.log

- Instead of appearing as popup windows the messages will appear in the console



Page heading

Page content

```
1                                                   script.js:2
2                                                   script.js:3
3                                                   script.js:4
4                                                   script.js:5
5                                                   script.js:6
1                                                   script.js:2
2                                                   script.js:3
3                                                   script.js:4
4                                                   script.js:5
5                                                   script.js:6
>
```

Exercise 3

< 5 minutes

1) Replace alert() with console.log

2) Open the console and refresh the page

3) Make sure you can see where the numbers have been printed to!

Selecting elements in Javascript

- Javascript contains inbuilt functions for selecting HTML elements so you can change properties on the (css, attributes, etc)
- The simplest way is to give an element an ID in the HTML:

```
<!DOCTYPE html>
<html>
  <head>
    <title>My Web Page!</title>
    <script src="script.js"></script>
  </head>

  <body>
    <h1 id="pageheading">Page heading</h1>
    <p>Page content</p>
  </body>
</html>
```

Selecting elements with Javascript

- Once an element on the page has an ID, you can use the javascript function document.getElementById() to select it and store the *element* in a variable

```html
<!DOCTYPE html>
<html>
  <head>
    <title>My Web Page!</title>
     <script src="script.js"></script>
  </head>

  <body>
    <h1 id="pageheading">
         Page heading
     </h1>
    <p>Page content</p>
  </body>
</html>
```

```javascript
var element = document.getElementById('pageheading');
```

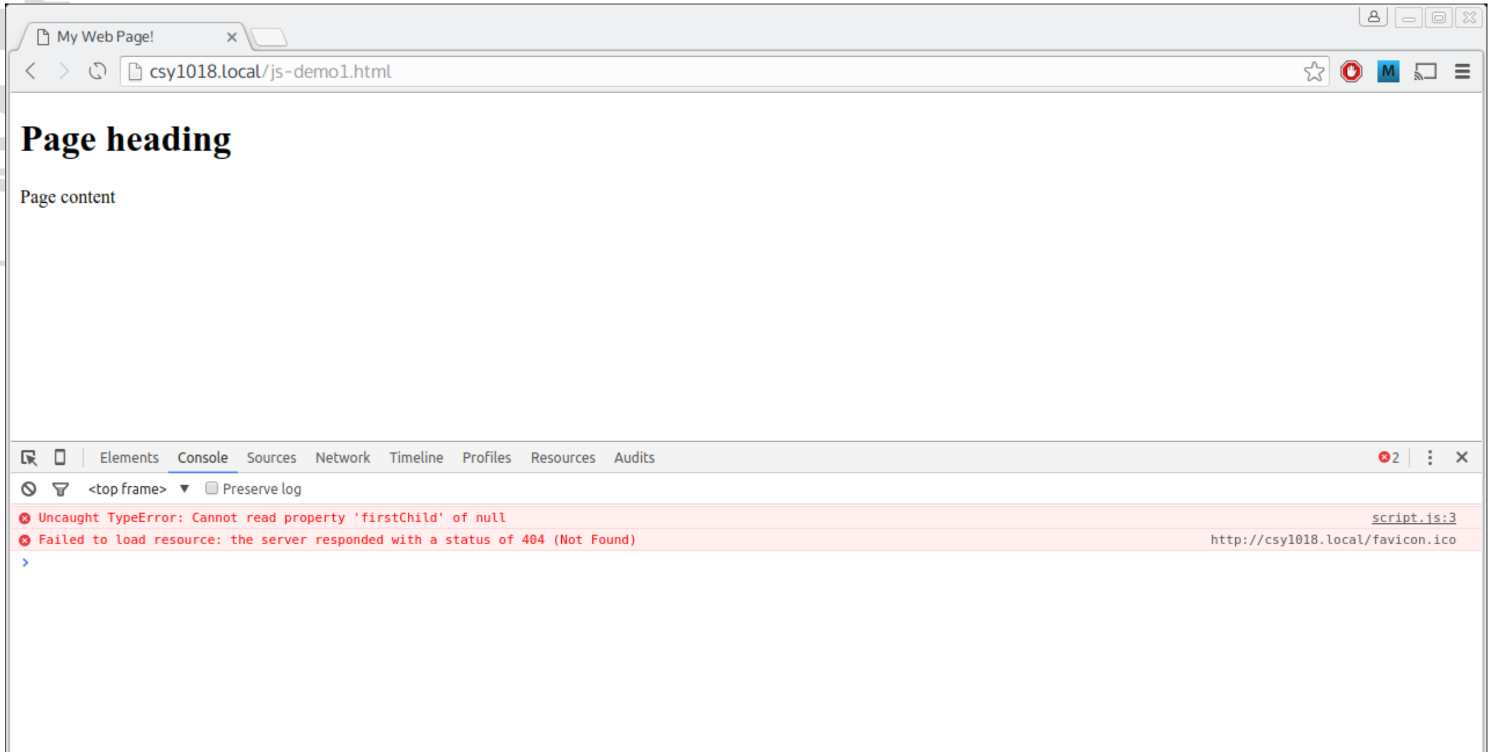
Selecting elements

- Once you have an element you can make changes to it
- E.g. to update the content you can use:

```
var element = document.getElementById('pageheading');
element.firstChild.nodeValue = 'New Heading';
```

Javascript - Basics

- Running this code won't quite have the desired effect!
- Hint: Always keep the console open as it will tell you if there are any errors in your code!



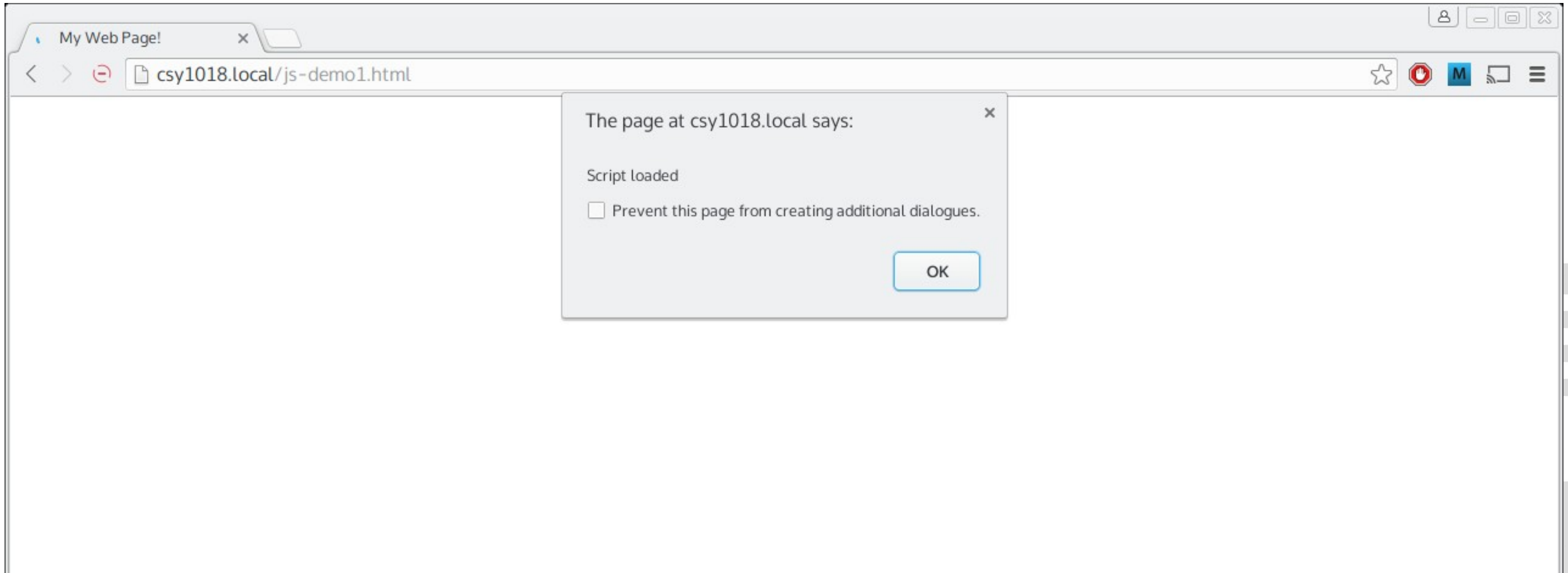
Page heading

Page content

```
Uncaught TypeError: Cannot read property 'firstChild' of null                                    script.js:3
Failed to load resource: the server responded with a status of 404 (Not Found)      http://csy1018.local/favicon.ico
```

Javascript

- Remember the first alert box()
- The Javascript code is run before any elements exist on the page, which is why the code is failing



The page at csy1018.local says:

Script loaded

Prevent this page from creating additional dialogues.

Javascript

- Rather than having the code run before the page has loaded, it's possible to write a function that is run when the page loads.
- This requires 2 steps:
  - 1) Move the code you want to run when the page loads into a function
  - 2) Inform the browser you want to run this function when the page loads

Javascript

1) Move the code you want to run when the page loads into a function

```
function myLoadFunction() {
        var element = document.getElementById('pageheading');

        element.firstChild.nodeValue = 'New Heading';
}
```

- Note: This function can have any name you like!

Javascript

2) Inform the browser you want to run this function when the page loads

– This is done using the inbuilt function document.addEventListener function

```
function myLoadFunction() {
        var element = document.getElementById('pageheading');
        element.firstChild.nodeValue = 'New Heading';
}


document.addEventListener('DOMContentLoaded', myLoadFunction);
```

DOMContentLoaded
means when the content on the
Page is loaded (the elements exist)

The name of the function

Javascript

- addEventListener is a very useful function
- It allows you to run a function when a specific *event* occurs

```
document.addEventListener('DOMContentLoaded', myLoadFunction);
```

When this happens

Run the function with this name

Exercise 4

5-10 mins

1) If you don't have one already add a <h1> element to your page

2) Give your <h1> element an ID

3) Using **addEventListener** update the content of the <h1> element on the page when it loads

4) Verify this works by testing it in your browser

5) Add another element to the page such as <p> and update the contents on page load the same way you did with the <h1>

6) Verify this works by testing it in your browser

Hint: You should only need one function and one line with addEventListener!

Exercise 4 - Solution

```
<!DOCTYPE html>
<html>
  <head>
    <title>My Web Page!</title>
     <script src=”script.js”></script>
  </head>

  <body>
    <h1 id=”pageheading”>
          Page heading
     </h1>
    <p id=”paragraph”>Page content</p>
  </body>
</html>
```

```
function myLoadFunction() {
        var element = document.getElementById('pageheading');
        element.firstChild.nodeValue = 'New Heading';

        var element = document.getElementById('paragraph');
        element.firstChild.nodeValue = 'New paragraph text';
}


document.addEventListener('DOMContentLoaded', myLoadFunction);
```

Javascript

- There is also a 'click' event which occurs whenever the element is clicked on

```
document.addEventListener('click', myClickFunction);
```

- This will run `myClickFunction` whenever the *document* is clicked on (the document is the entire page)

Exercise 5

- **< 5 mins**
- 1) Change DOMContentLoaded to click
- 2) Refresh the page – the <h1> and <p> elements should start with one value and change when you click somewhere on the page!

Exercise 5 - Solution

```
<!DOCTYPE html>
<html>
  <head>
    <title>My Web Page!</title>
     <script src=”script.js”></script>
  </head>

  <body>
    <h1 id=”pageheading”>
         Page heading
     </h1>
    <p id=”paragraph”>Page content</p>
  </body>
</html>
```

```
function myLoadFunction() {
        var element = document.getElementById('pageheading');
        element.firstChild.nodeValue = 'New Heading';

        var element = document.getElementById('paragraph');
        element.firstChild.nodeValue = 'New paragraph text';
}


document.addEventListener('click', myLoadFunction);
```

Click events

- If you click **anywhere** on the document, the contents of both elements will be updated
- It's possible to assign a click event to a particular element
- However, the code that associates the event with the element also has to be done **after** the page has loaded

Click Events

- You can call *element.addEventListener* to add an event to a specific element
- This works exactly the same way as document.addEventListener however it will only call your function when that particular element is clicked on

```
function myClickFunction() {
        var element = document.getElementById('pageheading');
        element.firstChild.nodeValue = 'New Heading';

         var element = document.getElementById('paragraph');
         element.firstChild.nodeValue = 'New paragraph text';
}

function myLoadFunction() {
        var element = document.getElementById('pageheading');
        element.addEventListener('click', myClickFunction);
}


document.addEventListener('DOMContentLoaded', myLoadFunction);
```

The click function updates the contents of the h1 and p elements

When the page loads, run myLoadFunction

When the load function is run, a click event is added to the h1 Element so that myClickFunction is run when the h1 is clicked on

Exercise 6

5 – 10 minutes

1) Make it so that when the h1 is clicked, the contents of the h1 element is update (and not the contents of the paragraph)

2) Make it so that when the p element is clicked, the contents of the p element is update (and not the contents of the h1)

3) Test out other event types to see how they work:

- mouseenter
- mouseleave

4) Multiple event listeners can be applied to the same element, amend the program so that the text is changed when you mouse over the element and it reverts to the original text when the mouse exits the element

Exercise 6 - Solutions

```
function updateParagraph() {
        var element = document.getElementById('paragraph');
           element.firstChild.nodeValue = 'New Heading';
}

function updateH1() {
        var element = document.getElementById('pageheading');
           element.firstChild.nodeValue = 'New paragraph contents';
}

function myLoadFunction() {
        var element = document.getElementById('pageheading');
        element.addEventListener('click', updateH1);

        var element = document.getElementById('paragraph');
        element.addEventListener('click', updateParagraph);
}


document.addEventListener('DOMContentLoaded', myLoadFunction);
```

Update the contents
Of the <p> element

Update the contents
Of the <h1> element

When the H1
is clicked,
run the function
updateH1

When the paragraph
is clicked,
run the function
updateParagraph

When the page loads,
run myLoadFunction

Forms

- When using <input> or <textarea> elements you can read the contents of the box (the value the user has typed in) using the .value property
-

```
<input type=”text” id=”myinput” />
```

```
var element = document.getElementById('myinput');
alert(element.value);
```

Exercise 7

- 10-15 minutes
- 1) Create a new HTML page with the following elements:
  - 1 text box (<input type=”text” />)
  - 1 button (<input type=”button” />)
- 2) When the button is pressed display the value in the text box in an alert
- 3) Add a <div> element to the page
- 4) When the button is pressed, instead of displaying the contents of the input in the alert, display it in the created <div>
- 5) Remove the button from the page. There is a \`keyup\` event which can be assigned to input elements, use this to update the div each time the contents of the input box changes

Exercise 7 Solutions

```
<!DOCTYPE html>
<html>
  <head>
    <title>My Web Page!</title>
        <script src="script.js"></script>
  </head>

  <body>
    <input type="text" id="myinput" />
    <input type="button" id="mybutton" value="click me" />
  </body>
</html>
```

```
function clickFunction() {
        var element = document.getElementById('myinput');
        alert(element.value);
}

function myLoadFunction() {
        var element = document.getElementById('mybutton');
        element.addEventListener('click', clickFunction);
}

document.addEventListener('DOMContentLoaded', myLoadFunction);
```

Exercise 7 Solutions

```html
<!DOCTYPE html>
<html>
  <head>
    <title>My Web Page!</title>
        <script src="script.js"></script>
  </head>

  <body>
    <input type="text" id="myinput" />
    <input type="button" id="mybutton" value="click me" />
    <div id="result">  </div>
  </body>
</html>
```

```javascript
function clickFunction() {
        var element = document.getElementById('myinput');
        var div = document.getElementById('result');
        div.firstChild.nodeValue = element.value;
}

function myLoadFunction() {
        var element = document.getElementById('mybutton');
        element.addEventListener('click', clickFunction);

}

document.addEventListener('DOMContentLoaded', myLoadFunction);
```

Exercise 7 Solutions

```html
<!DOCTYPE html>
<html>
  <head>
    <title>My Web Page!</title>
        <script src="script.js"></script>
  </head>

  <body>
    <input type="text" id="myinput" />
    <div id="result">  </div>
  </body>
</html>
```

```javascript
function clickFunction() {
        var element = document.getElementById('myinput');
        var div = document.getElementById('result');
        div.firstChild.nodeValue = element.value;
}

function myLoadFunction() {
        var element = document.getElementById('myinput');
        element.addEventListener('keyup', clickFunction);

}

document.addEventListener('DOMContentLoaded', myLoadFunction);
```