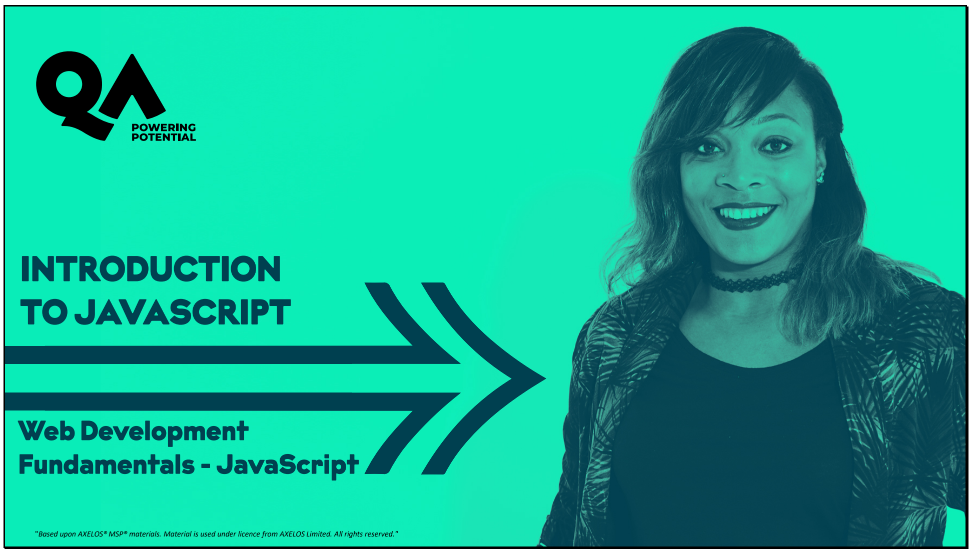
Slide 1

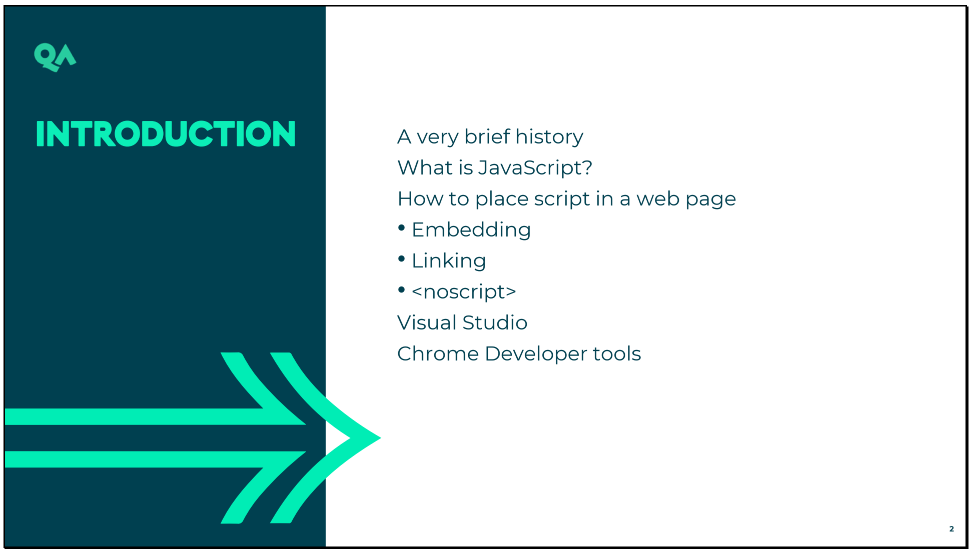


POWERING POTENTIAL

INTRODUCTION TO JAVASCRIPT

Web Development Fundamentals - JavaScript

Slide 2

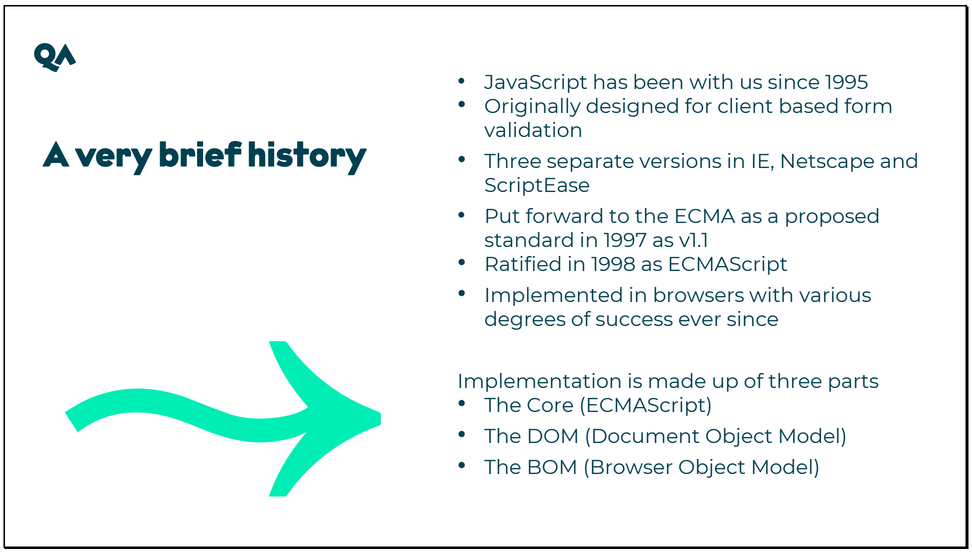


INTRODUCTION

A very brief history
What is JavaScript?
How to place script in a web page
- Embedding
- Linking
- <noscript>

Visual Studio
Chrome Developer tools

Slide 3



A very brief history

- JavaScript has been with us since 1995
- Originally designed for client based form validation
- Three separate versions in IE, Netscape and ScriptEase
- Put forward to the ECMA as a proposed standard in 1997 as v1.1
- Ratified in 1998 as ECMAScript
- Implemented in browsers with various degrees of success ever since

Implementation is made up of three parts
- The Core (ECMAScript)
- The DOM (Document Object Model)
- The BOM (Browser Object Model)

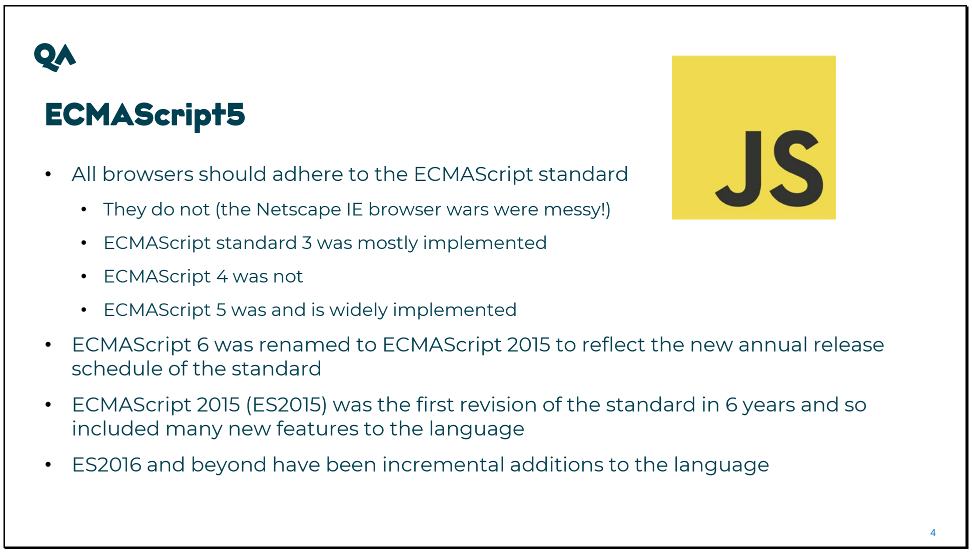
Whether you are here as a new web programmer or an old hand server guru forced to consider the front end in a different way because of those dashed, new fangled MVC approaches, JavaScript is a core part of any web developer's arsenal. JavaScript is supported in almost every browser today and increasingly in a universal way; but, as we will see, older browsers cause us real issues with the way we code and what we can do.

JavaScript is a prototype-based scripting language that is dynamic, weakly typed and has first-class functions. It is a multi-paradigm language, supporting object-oriented, imperative, and functional programming styles.

JavaScript was formalised in the ECMAScript language standard and is primarily used in the form of client-side JavaScript, implemented as part of a Web browser in order to provide enhanced user interfaces and dynamic websites. This enables programmatic access to computational objects within a host environment.

In recent years, it has found usage in non-web programming architectures, such as Acrobat files and Windows 8 Metro applications, demonstrating the amazing versatility of this incredibly flexible and powerful language and server-side programming with tools such as Node.js.

Slide 4



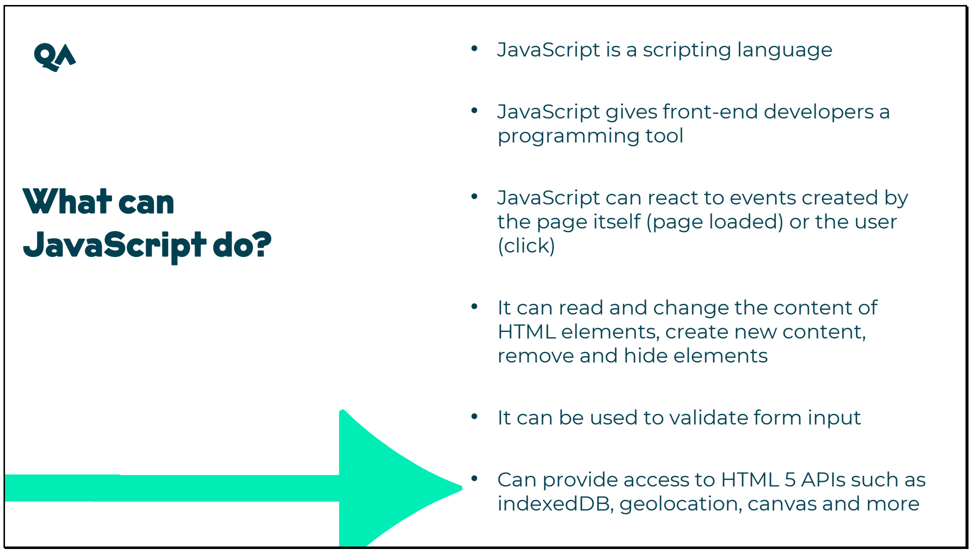
ECMAScript5

- All browsers should adhere to the ECMAScript standard
  - They do not (the Netscape IE browser wars were messy!)
  - ECMAScript standard 3 was mostly implemented
  - ECMAScript 4 was not
  - ECMAScript 5 was and is widely implemented
- ECMAScript 6 was renamed to ECMAScript 2015 to reflect the new annual release schedule of the standard
- ECMAScript 2015 (ES2015) was the first revision of the standard in 6 years and so included many new features to the language
- ES2016 and beyond have been incremental additions to the language

4

The 6th edition, officially known as ECMAScript 2015, was finalised in June 2015. This update adds significant new syntax for writing complex applications, including classes and modules, but defines them semantically in the same terms as ECMAScript 5 strict mode. Other new features include iterators and for/of loops, Python-style generators and generator expressions, arrow functions, binary data, typed arrays, collections (maps, sets and weak maps), promises, number and math enhancements, reflection, and proxies (metaprogramming for virtual objects and wrappers).

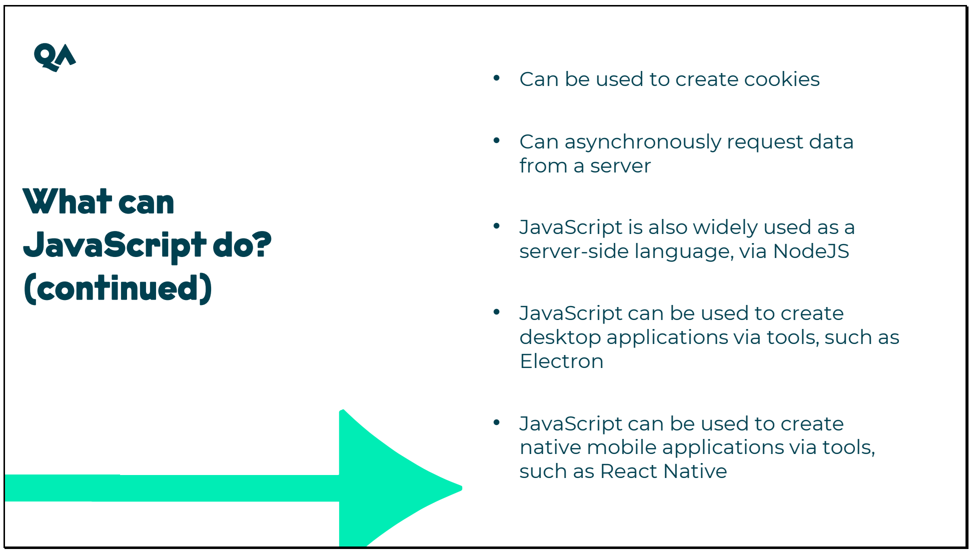
Slide 5



What can JavaScript do?

- JavaScript is a scripting language
- JavaScript gives front-end developers a programming tool
- JavaScript can react to events created by the page itself (page loaded) or the user (click)
- It can read and change the content of HTML elements, create new content, remove and hide elements
- It can be used to validate form input
- Can provide access to HTML 5 APIs such as indexedDB, geolocation, canvas and more

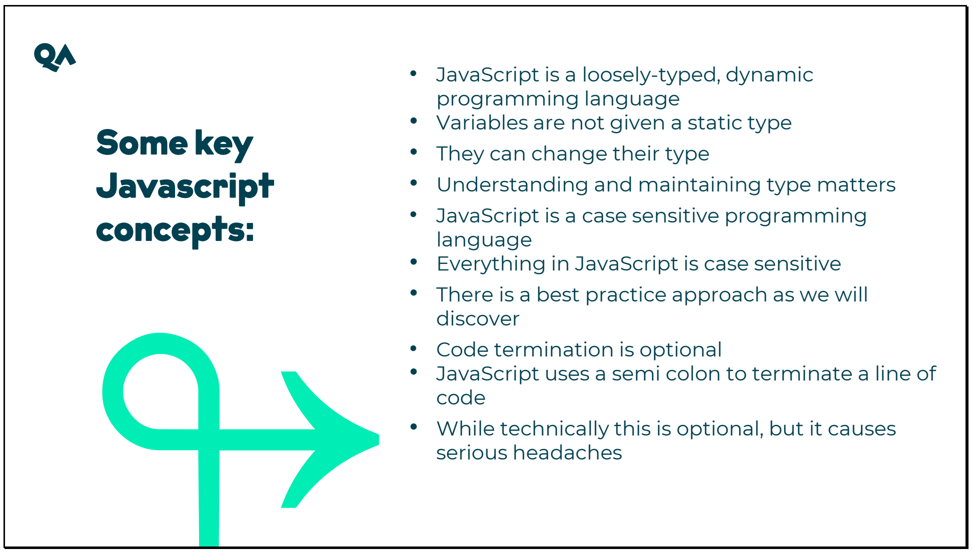
Slide 6



What can JavaScript do? (continued)

- Can be used to create cookies
- Can asynchronously request data from a server
- JavaScript is also widely used as a server-side language, via NodeJS
- JavaScript can be used to create desktop applications via tools, such as Electron
- JavaScript can be used to create native mobile applications via tools, such as React Native

Slide 7



Some key Javascript concepts:

- JavaScript is a loosely-typed, dynamic programming language
- Variables are not given a static type
- They can change their type
- Understanding and maintaining type matters
- JavaScript is a case sensitive programming language
- Everything in JavaScript is case sensitive
- There is a best practice approach as we will discover
- Code termination is optional
- JavaScript uses a semi colon to terminate a line of code
- While technically this is optional, but it causes serious headaches

As we begin our journey with JavaScript, we will get a few things locked down in our brainboxes that will save us a lot of heartache and pain as the course goes on.

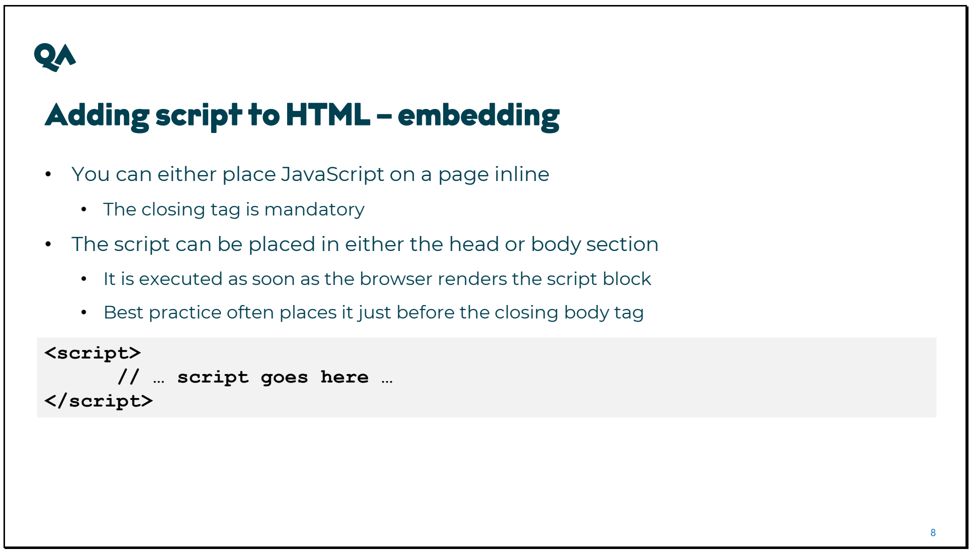
JavaScript is a case-sensitive language. This means…

```
let numberOne = 5;
let NumberOne = 5;
```

… actually creates two separate variables! This is a key fact to remember as you code – be careful as we move on in the development of our code.

JavaScript will work without a semicolon terminating the code in many situations, but this will also cause you real issues as the course draws to a conclusion and we discuss minificiation of script. Please try, therefore, to remember that every instruction needs to be terminated.

Slide 8



Adding script to HTML – embedding

- You can either place JavaScript on a page inline
  - The closing tag is mandatory
- The script can be placed in either the head or body section
  - It is executed as soon as the browser renders the script block
  - Best practice often places it just before the closing body tag

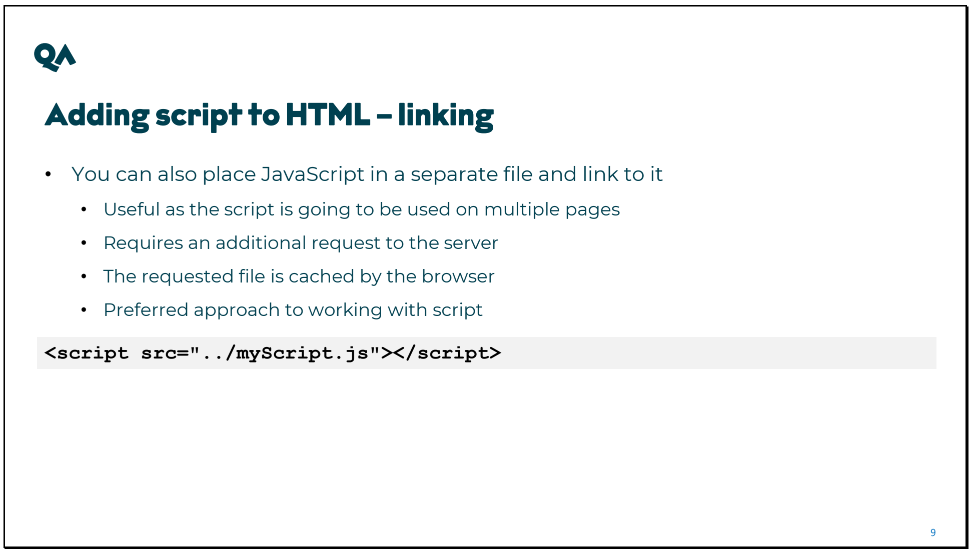
```
<script>
      // … script goes here …
</script>
```

8

The browser needs to know the data it is rendering is not just normal text and is actually JavaScript. We achieve this by using the <script> element. You may use as many script elements as you like on a page. Browsers today will assume that your script is client side JavaScript, if no further information is provided.

Script blocks are executed as soon as the browser reaches the code block in the page. The script will then be executed; we will see the importance of this concept as the course progresses.

Slide 9



Adding script to HTML – linking

- You can also place JavaScript in a separate file and link to it
  - Useful as the script is going to be used on multiple pages
  - Requires an additional request to the server
  - The requested file is cached by the browser
  - Preferred approach to working with script

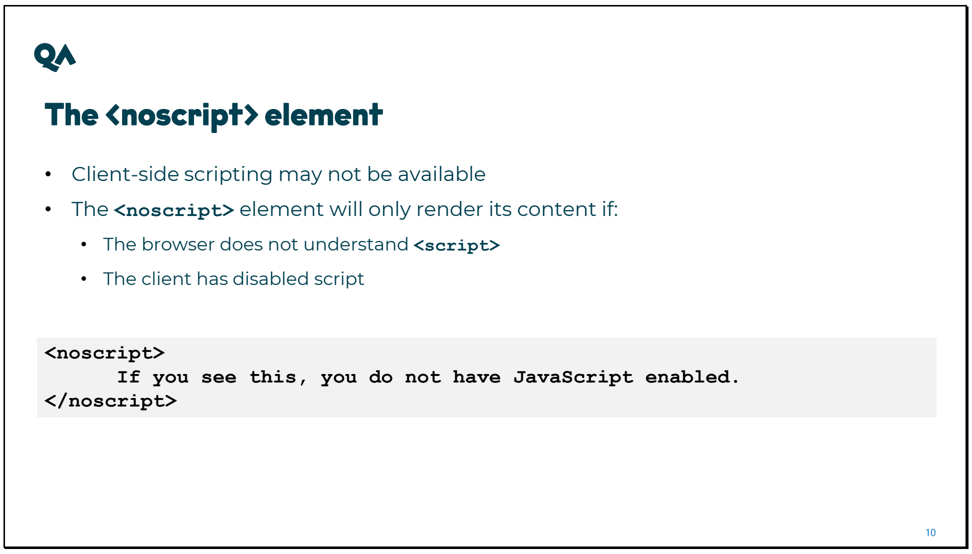
```
<script src="../myScript.js"></script>
```

9

Everything said about inline script processing is also true for linked scripts. The key consideration when using external script references is with performance. Each script file is a separate request to the server for the file. On first request to the page, this can cause some lag in a page rendering. Older browser, IE 6 being a notable example, can only make two asynchronous file requests from a server at a time, so the page load can be uneven. Very often, you compact your files together to create a single script file to assist with this; we will see this strategy later in the course.

The big benefit of this strategy of external linking is that script to be used on multiple pages can be managed centrally, making site maintenance much simpler and, as per usual, with asset caching the initial first cost leads to a quicker load on subsequent visits.

Slide 10

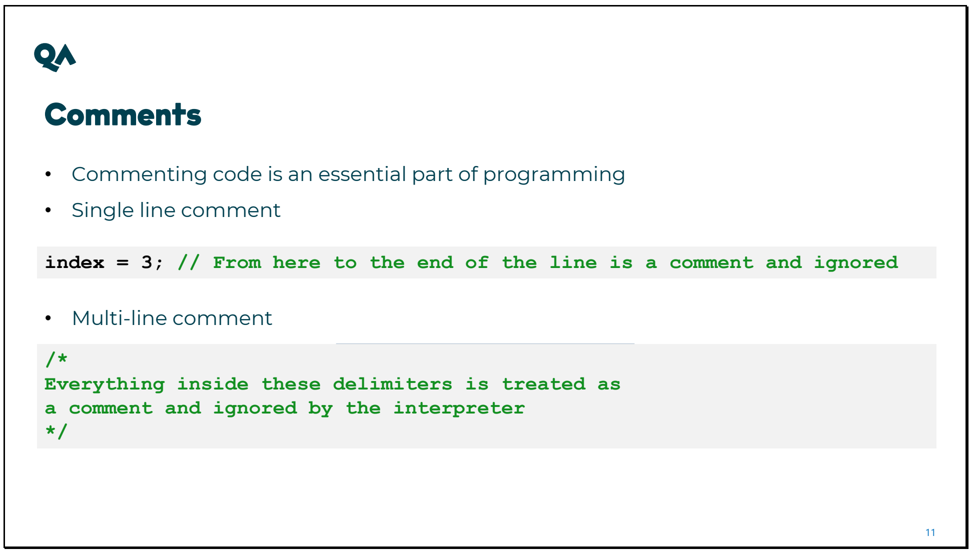


The <noscript> element

- Client-side scripting may not be available
- The **`<noscript>`** element will only render its content if:
  - The browser does not understand **`<script>`**
  - The client has disabled script

```
<noscript>
      If you see this, you do not have JavaScript enabled.
</noscript>
```

Slide 11



Comments

- Commenting code is an essential part of programming
- Single line comment

```
index = 3; // From here to the end of the line is a comment and ignored
```

- Multi-line comment

```
/*
Everything inside these delimiters is treated as
a comment and ignored by the interpreter
*/
```

11

JavaScript has two ways of marking a comment: // starts a comment, which ends at the end of the current line; and multi-line comments, which can be created by inserting the comment text between /* (start-comment) and */ (end-comment).

Note that multi-line comments do not nest. This typical debugging method will cause problems:

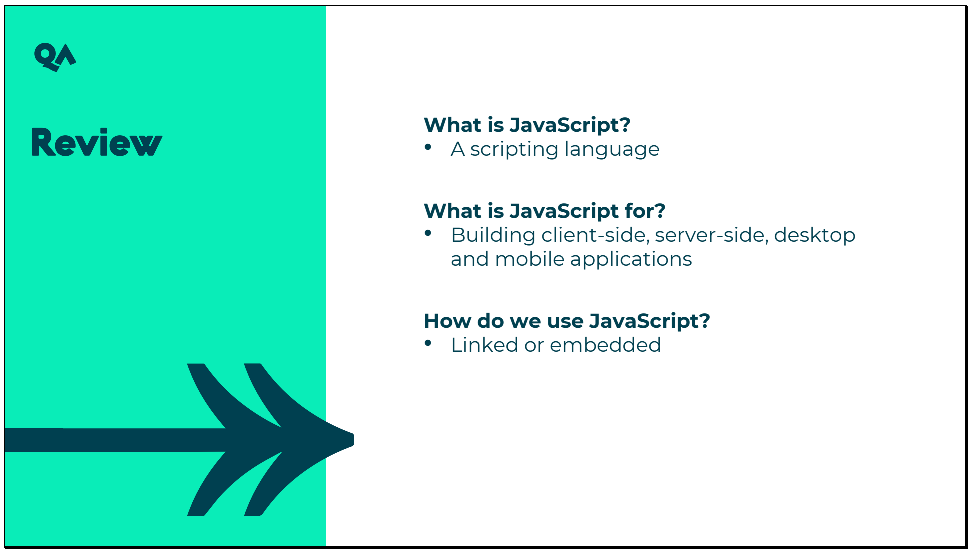
```
/*   something wrong in this function
     commented out temporarily to isolate the problem

     function fred()
     {
        /* this function computes mortgage interest
        */
        x = ((y * 3) + 4 )/ 6.291 ^ 3.412
     }

*/
```

The first */ will close all comments. The second */ will be a syntax error.

Slide 12



Review

What is JavaScript?
- A scripting language

What is JavaScript for?
- Building client-side, server-side, desktop and mobile applications

How do we use JavaScript?
- Linked or embedded