

INTRODUCTION TO JAVASCRIPT DEVELOPMENT

- The Magic of Dynamic Web Pages



Nikolay Chochev

- Technical Trainer
- Chochev.EU Team
- <http://chochev.eu/web/>



TABLE OF CONTENTS

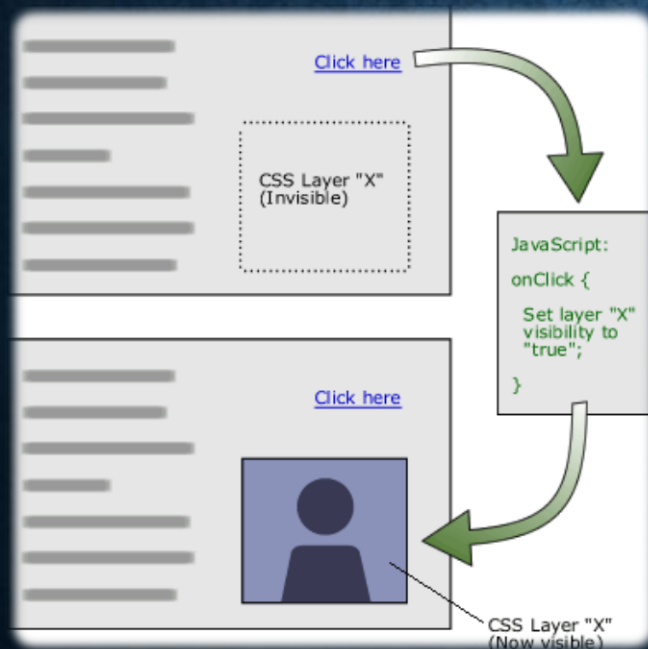
- Dynamic HTML
- How to Create DHTML?
 - XHTML, CSS, JavaScript, DOM
- Intro to JavaScript
 - History
 - JavaScript in Web Pages
- JavaScript Syntax
- Pop-up boxes
- Debugging in JavaScript





DYNAMIC HTML

Dynamic Behavior at the Client Side



WHAT IS DHTML?

- Dynamic HTML (DHTML)
 - Makes possible a Web page to react and change in response to the user's actions
- DHTML consists of HTML + CSS + JavaScript



DHTML = HTML + CSS + JAVASCRIPT

- HTML defines Web sites content through semantic tags (headings, paragraphs, lists, ...)
- CSS defines 'rules' or 'styles' for presenting every aspect of an HTML document
 - Font (family, size, color, weight, etc.)
 - Background (color, image, position, repeat)
 - Position and layout (of any object on the page)
- JavaScript defines dynamic behavior
 - Programming logic for interaction with the user, to handle events, etc.



JAVASCRIPT

Dynamic Behavior in a Web Page

JAVASCRIPT

- JavaScript is a front-end scripting language developed by Netscape for dynamic content
 - Lightweight, but with limited capabilities
 - Can be used as object-oriented language
 - Embedded in your HTML page
 - Interpreted by the Web browser
- Client-side, mobile and desktop technology
- Simple and flexible
- Powerful to manipulate the DOM

JAVASCRIPT ADVANTAGES

- JavaScript allows interactivity such as:
 - Implementing form validation
 - React to user actions, e.g. handle keys
 - Changing an image on moving mouse over it
 - Sections of a page appearing and disappearing
 - Content loading and changing dynamically
 - Performing complex calculations
 - Custom HTML controls, e.g. scrollable table
 - Implementing AJAX functionality

WHAT CAN JAVASCRIPT DO?

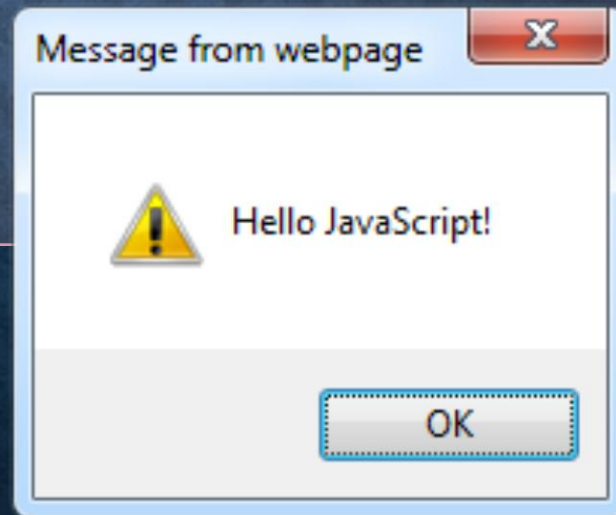
- Can handle events
- Can read and write HTML elements and modify the DOM tree
- Can validate form data
- Can access / modify browser cookies
- Can detect the user's browser and OS
- Can be used as object-oriented language
- Can handle exceptions
- Can perform asynchronous server calls (AJAX)

THE FIRST SCRIPT

```
<html>

<body>
  <script type="text/javascript">
    alert('Hello JavaScript!');
  </script>
</body>

</html>
```



FIRST JAVASCRIPT

Live Demo

USING JAVASCRIPT CODE

- The JavaScript code can be placed in:
 - `<script>` tag in the head
 - `<script>` tag in the body - not recommended
 - External files, linked via `<script>` tag the head
 - Files usually have `.js` extension
- Highly recommended
- The `.js` files get cached by the browser

```
<script src="scripts.js" type="text/javascript">  
<!-- code placed here will not be executed! -->  
</script>
```

JAVASCRIPT – WHEN IS EXECUTED?

- JavaScript code is executed during the page loading or when the browser fires an event
 - All statements are executed at page loading
 - Some statements just define functions that can be called later
- Function calls or code can be attached as "event handlers" via tag attributes
 - Executed when the event is fired by the browser

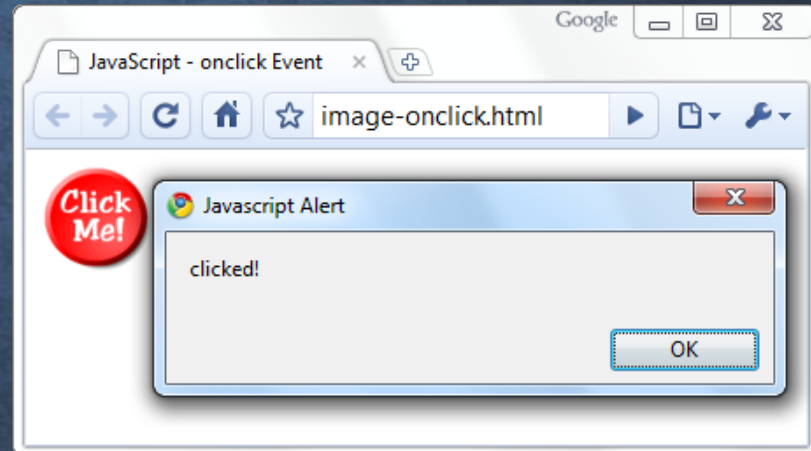
```

```

CALLING A JAVASCRIPT FUNCTION FROM EVENT HANDLER – EXAMPLE

```
<html>
<head>
<script type="text/javascript">
    function test (message) {
        alert(message);
    }
</script>
</head>

<body>
    
</body>
</html>
```



EVENT HANDLERS

Live Demo

USING EXTERNAL SCRIPT FILES

- Using external script files:

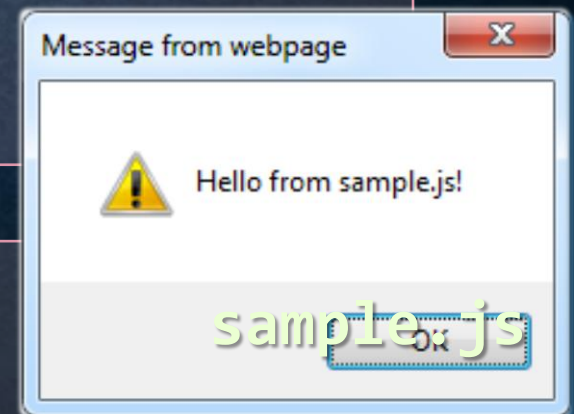
**external-
JavaScript.html**

```
<html>
<head>
  <script src="sample.js" type="text/javascript">
  </script>
</head>
<body>
  <button onclick="sample()" value="Call JavaScript
    function from sample.js" />
</body>
</html>
```

**The <script> tag is always
empty.**

- External JavaScript file:

```
function sample() {
  alert('Hello from sample.js!')
}
```



EXTERNAL JAVASCRIPT FILES

Live Demo

THE JAVASCRIPT SYNTAX

```
if (pop < 10)
{
    map.graphics.add(features[i].setSymbol(onePopSymbol));
}
else if (pop >= 10 && pop < 95)
{
    map.graphics.add(features[i].setSymbol(twoPopSymbol));
}
else if (pop >= 95 && pop < 365)
{
    map.graphics.add(features[i].setSymbol(threePopSymbol));
}
else if (pop >= 365 && pop < 1100)
{
    map.graphics.add(features[i].setSymbol(fourPopSymbol));
}
else
{
    map.graphics.add(features[i].setSymbol(fivePopSymbol));
}
```

JAVA
SCRIPT

JAVASCRIPT SYNTAX

- The JavaScript syntax is similar to C#
 - Operators (+, *, =, !=, &&, ++, ...)
 - Variables (typeless)
 - Conditional statements (if, else)
 - Loops (for, while)
 - Arrays (my_array[]) and associative arrays (my_array['abc'])
 - Functions (can return value)
 - Function variables (like the C# delegates)

STANDARD POPUP BOXES

- Alert box with text and [OK] button
 - Just a message shown in a dialog box:

```
alert("Some text here");
```

- Confirmation box
 - Contains text, [OK] button and [Cancel] button:

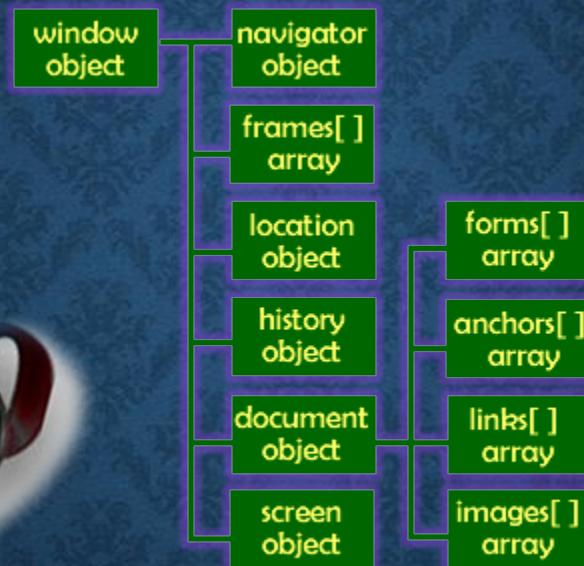
```
confirm("Are you sure?");
```

- Prompt box
 - Contains text, input field with default value:

```
prompt ("enter amount", 10);
```

POPUP BOXES

Live Demo

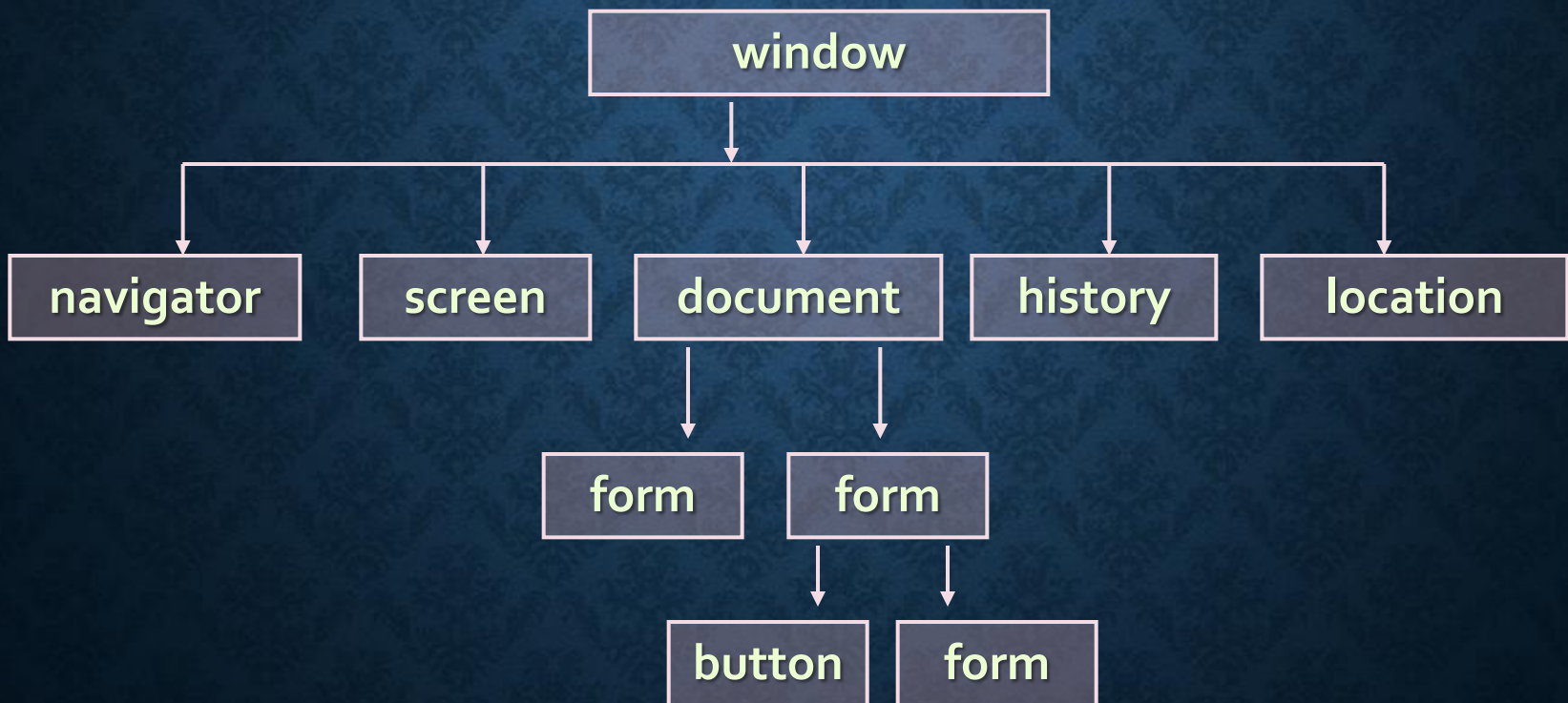


THE BUILT-IN BROWSER OBJECTS

BUILT-IN BROWSER OBJECTS

- The browser provides some read-only data via:
 - `window`
 - The top node of the DOM tree
 - Represents the browser's window
 - `document`
 - holds information the current loaded document
 - `screen`
 - Holds the user's display properties
 - `browser`
 - Holds information about the browser

DOM HIERARCHY – EXAMPLE



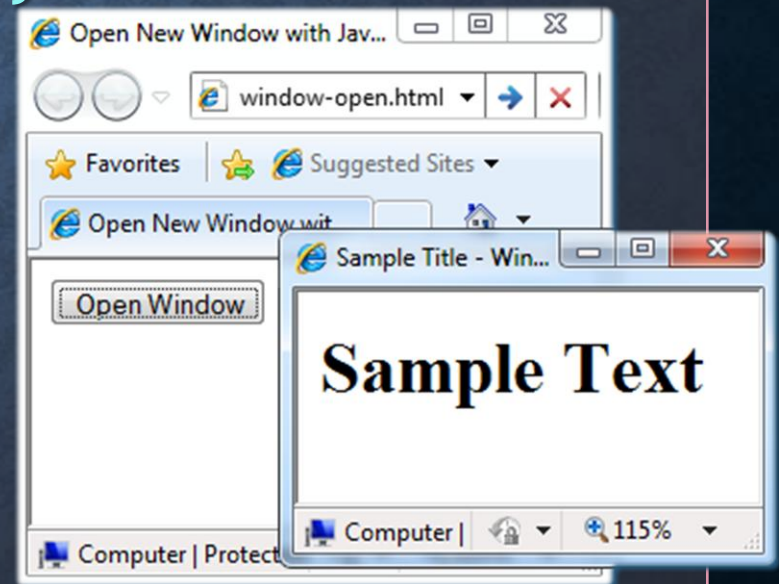
OPENING NEW WINDOW – EXAMPLE

window-open.html

- `window.open()`

```
var newWindow = window.open("", "sampleWindow",  
    "width=300, height=100, menubar=yes,  
    status=yes, resizable=yes");
```

```
newWindow.document.write(  
    "<html><head><title>  
    Sample Title</title>  
    </head><body><h1>Sample  
    Text</h1></body>");  
newWindow.status =  
    "Hello folks";
```



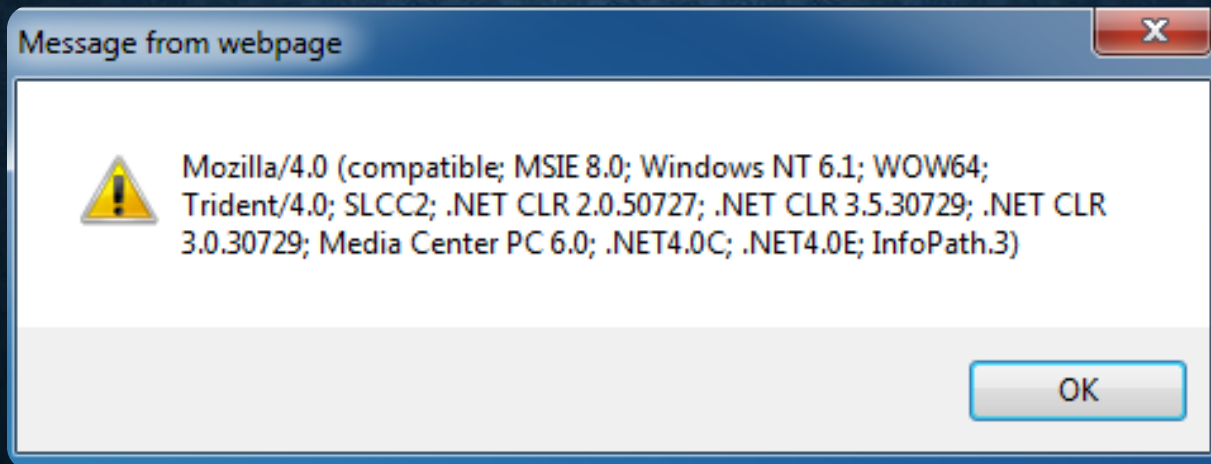
THE NAVIGATOR OBJECT

```
alert(window.navigator.userAgent);
```

The browser window

The navigator in the browser window

The userAgent (browser ID)



BUILT-IN BROWSER OBJECTS

Live Demo

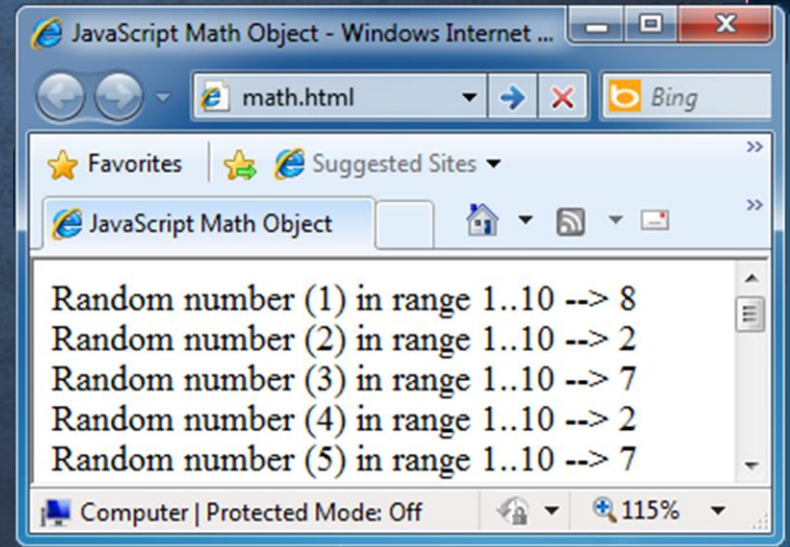
OTHER JAVASCRIPT OBJECTS

THE MATH OBJECT

math.html

- The Math object provides some mathematical functions

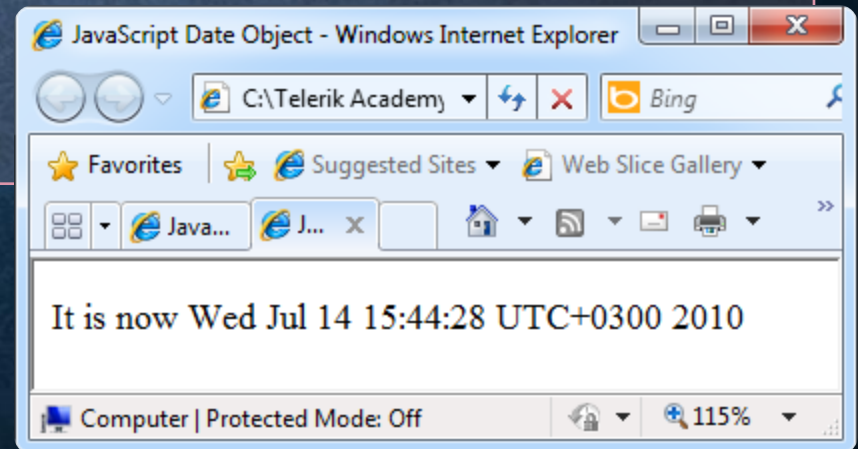
```
for (i=1; i<=20; i++) {  
    var x = Math.random();  
    x = 10*x + 1;  
    x = Math.floor(x);  
    document.write(  
        "Random number (" +  
        i + ") in range " +  
        "1..10 --> " + x +  
        "<br/>");  
}
```



THE DATE OBJECT

- The Date object provides date / calendar functions [dates.html](#)

```
var now = new Date();  
var result = "It is now " + now;  
document.getElementById("timeField")  
    .innerText = result;  
...  
<p id="timeField"></p>
```

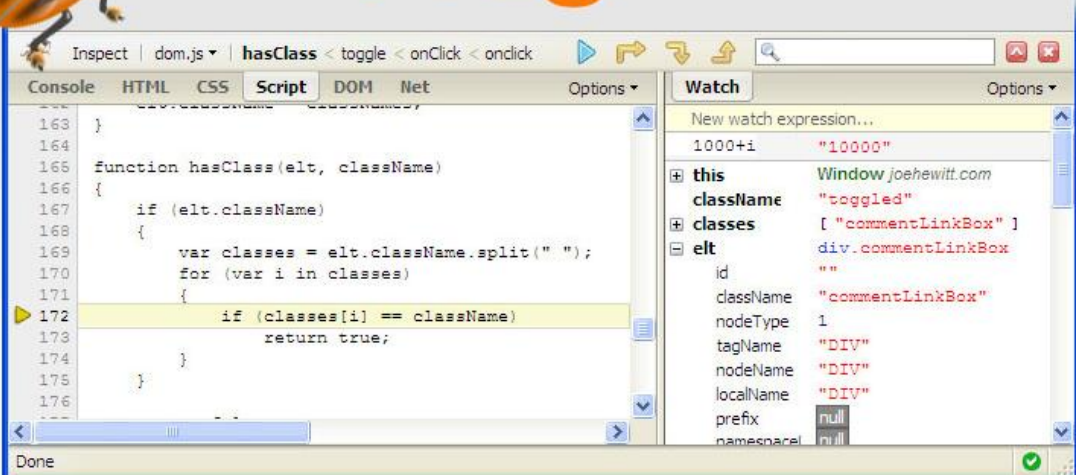


DEBUGGING JAVASCRIPT



blog
discuss
contribute
documentation

Firebug web development evolved



```
163 }
164
165 function hasClass(elt, className)
166 {
167     if (elt.className)
168     {
169         var classes = elt.className.split(" ");
170         for (var i in classes)
171         {
172             if (classes[i] == className)
173                 return true;
174         }
175     }
176 }
```

Done

JavaScript Debugging

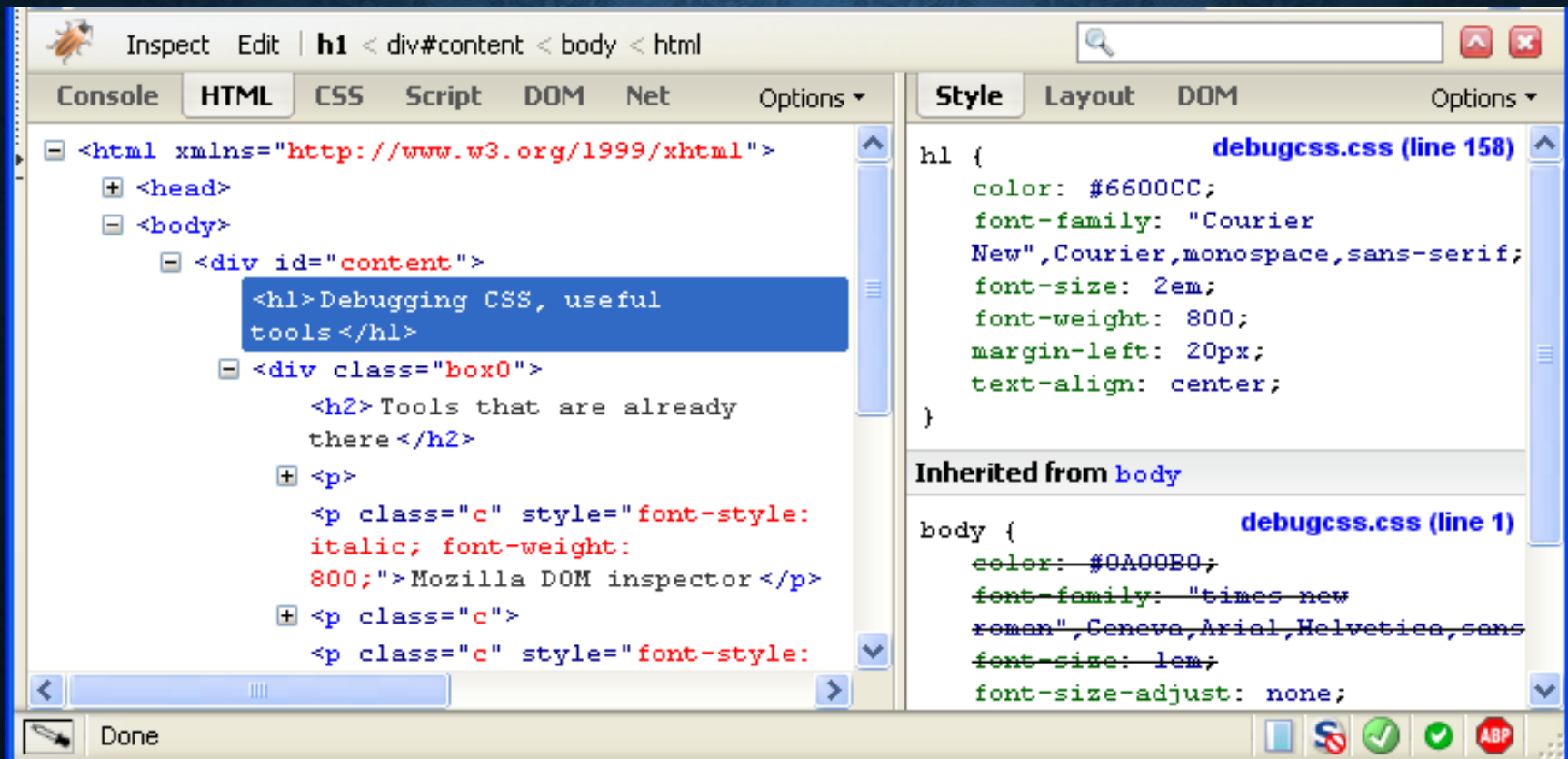
DEBUGGING JAVASCRIPT

- Modern browsers have JavaScript console where errors in scripts are reported
 - Errors may differ across browsers
- Several tools to debug JavaScript
 - Microsoft Script Editor
 - Add-on for Internet Explorer
 - Supports breakpoints, watches
 - JavaScript statement debugger; opens the script editor

FIREBUG

- Firebug – Firefox add-on for debugging JavaScript, CSS, HTML
 - Supports breakpoints, watches, JavaScript console editor
 - Very useful for CSS and HTML too
 - You can edit all the document real-time: CSS, HTML, etc
 - Shows how CSS rules apply to element
 - Shows Ajax requests and responses
 - Firebug is written mostly in JavaScript

FIREBUG (2)

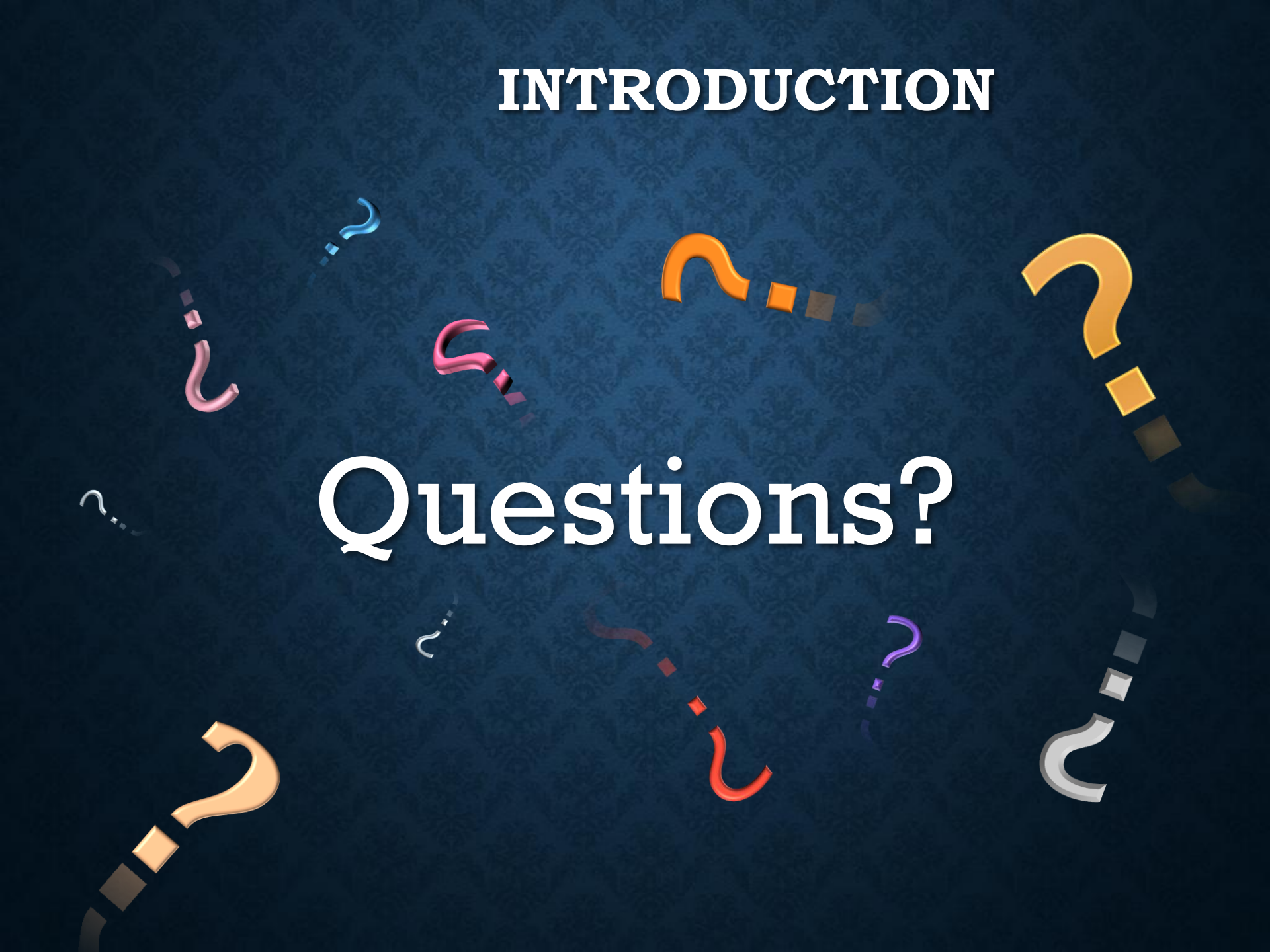


JAVASCRIPT CONSOLE OBJECT

- The `console` object exists only if there is a debugging tool that supports it
 - Used to write log messages at runtime
- Methods of the `console` object:
 - `debug(message)`
 - `info(message)`
 - `log(message)`
 - `warn(message)`
 - `error(message)`

INTRODUCTION

Questions?

The background is a dark blue gradient with a subtle, repeating pattern of small, light blue question marks. Scattered across the slide are several large, 3D-style question marks in various colors: orange, pink, blue, purple, red, and grey. These question marks are positioned around the central text, creating a visual theme of inquiry and questions.