Școala
informală
de IT

Introducere in IT

Programming (Javascript)

Programming

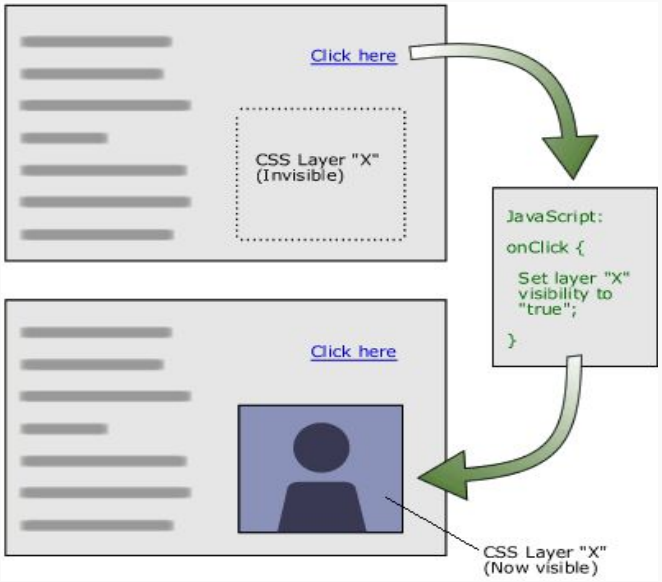
Agenda

- Introduction to Javascript
  - What is JavaScript
  - Implementing JavaScript into Web pages
- JavaScript Syntax

Introduction to JavaScript

What is DHTML?

DHTML
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 = 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)

What is 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
- Client-side technology
  - Embedded in your HTML page
  - Interpreted by the Web browser
- 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)

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

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

    - Highly recommended
    - The `.js` files get cached by the browser

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

```
<img src="logo.gif" onclick="alert('clicked!')" />
```

Calling a JavaScript Function from Event Handler - Example

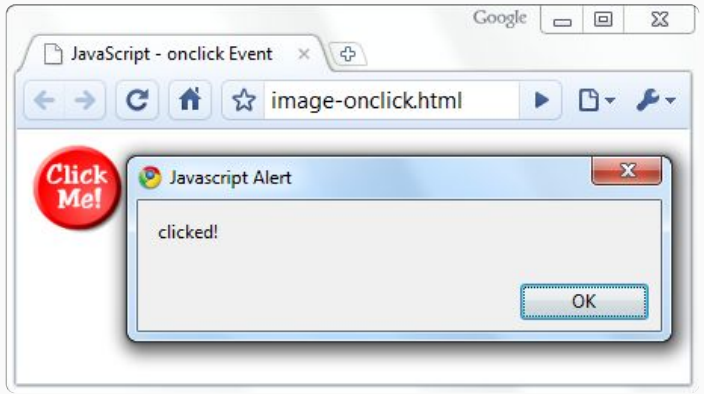
image-onclick.html

```
<html>
<head>
 <script type="text/javascript"
src="script.js"> </script>
</head>

<body>
  <img src="logo.gif"
    onclick="test('clicked!')"
/>
</body>
</html>
```

script.js

```
function test (message) {
    alert(message);
  }
```



The Javascript Syntax

JavaScript Syntax

- The JavaScript syntax is similar to C# and Java
  - 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)

Data Types

- JavaScript data types:
  - Numbers (integer, floating-point)
  - Boolean (true / false)
- String type – string of characters

```
var myName = "You can use both single or double quotes
for strings";
```

- Arrays and associative arrays (hash tables)

```
var myName = "You can use both single or double quotes
for strings";
```

- Undefined and null types

```
var someData;              // undefined
someData = null;
```

Everything is Object

- Every variable can be considered as object
  - For example strings and arrays have member functions:

```
var test = "some string";
alert(test[7]); // shows letter 'r'
alert(test.charAt(5)); // shows letter 's'
alert("test".charAt(1)); //shows letter 'e'
alert("test".substring(1,3)); //shows 'es'
```

```
var arr = [1,3,4];
alert (arr.length); // shows 3
arr.push(7); // appends 7 to end of array
alert (arr[3]); // shows 7
```

Control structures

Conditional Statement (if)

```
unitPrice = 1.30;
if (quantity > 100) {
  unitPrice = 1.20;
}
else {
  unitPrice = 1.40;
}
```

>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to
==	Equal
!=	Not equal

Conditional Statement (if)

- The condition may be of Boolean or integer type:

```
var a = 0;
var b = true;
if (typeof(a)=="undefined" || typeof(b)=="undefined")
{
  document.write("Variable a or b is undefined.");
}
else if (!a && b) {
  document.write("a==0; b==true;");
} else {
  document.write("a==" + a + "; b==" + b + ";");
}
```

Switch Statement

- The `switch` statement:

```
switch (variable) {
  case 1:
    // do something
    break;
  case 'a':
    // do something else
    break;
  case 3.14:
    // another code
    break;
  default:
    // something completely different
}
```

Loops

- `for` loop
- `while` loop
- `do ... while` loop

```
for (var counter = 0; counter < 4; counter++) {
  alert(counter);
}
```

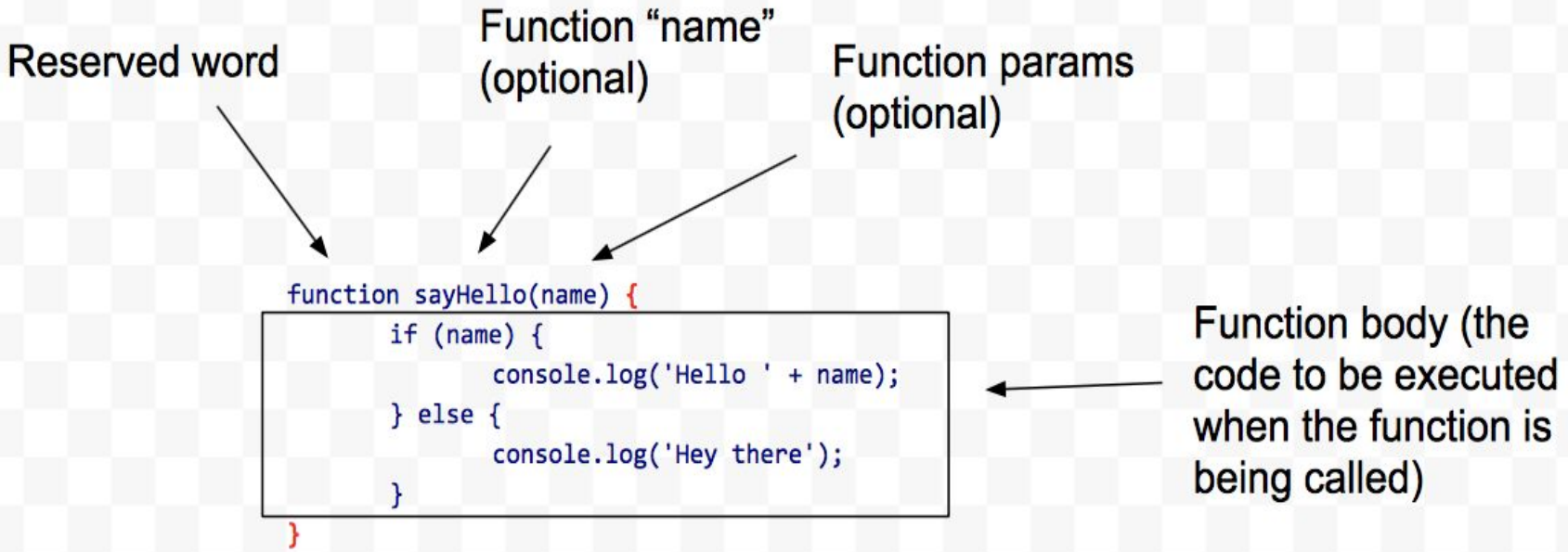
```
var counter = 0;
while (counter < 4) {
  alert(counter);
  counter++;
}
```

Functions

Functions

- **= pieces of programs made of of statements that perform a task or calculate a value**
- **Why they are useful:**
  - **reuse code / reduce repetition**
    - You can use the same code many times with different arguments, to produce different results.
  - **structure larger programs**
  - **associate names with subprograms**
  - **isolate these subprograms from each other**

Functions


Reserved word
Function "name" (optional)
Function params (optional)
function sayHello(name) {
if (name) {
console.log('Hello ' + name);
} else {
console.log('Hey there');
}
}
Function body (the code to be executed when the function is being called)

Functions

- Code structure – splitting code into parts
- Data comes in, processed, result returned

```
function average(a, b, c)
{
    var total;
    total = a+b+c;
    return total/3;
}
```

Parameters come in here

Declaring variables is optional. Type is never declared

Value returned here

Working with functions & operations

String Operations

- The + operator joins strings

```
string1 = "fat ";
string2 = "cats";
alert(string1 + string2);  // fat cats
```

- What is "9" + 9?

```
alert("9" + 9);  // 99
```

- Converting string to number:

```
alert(parseInt("9") + 9);  // 18
```

Arrays Operations and Properties

- Declaring new empty array:

```
var arr = new Array();
```

- Declaring an array holding few elements:

```
var arr = [1, 2, 3, 4, 5];
```

- Appending an element / getting the last element:

```
arr.push(3);
var element = arr.pop();
```

- Reading the number of elements (array length):

```
arr.length;
```

- Finding element's index in the array:

```
arr.indexOf(1);
```

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);
```

Sum of Numbers - Example

sum.html

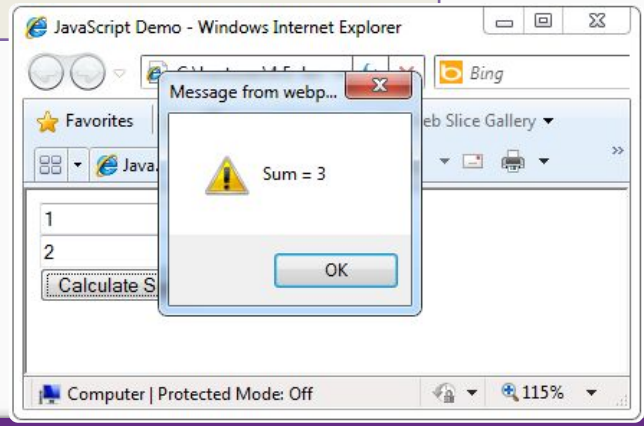
```
<html>
<head>
  <title>JavaScript Demo</title>
  <script type="text/javascript" src="script.js"> </script>
</head>

<body>
  <form name="mainForm">
    <input type="text" name="textBox1" /> <br/>
    <input type="text" name="textBox2" /> <br/>
    <input type="button" value="Process"
      onclick="javascript: calcSum()" />
    <input type="text" name="textBoxSum"
      readonly="readonly"/>
  </form>
</body>
</html>
```

Sum of Numbers - Example (2)

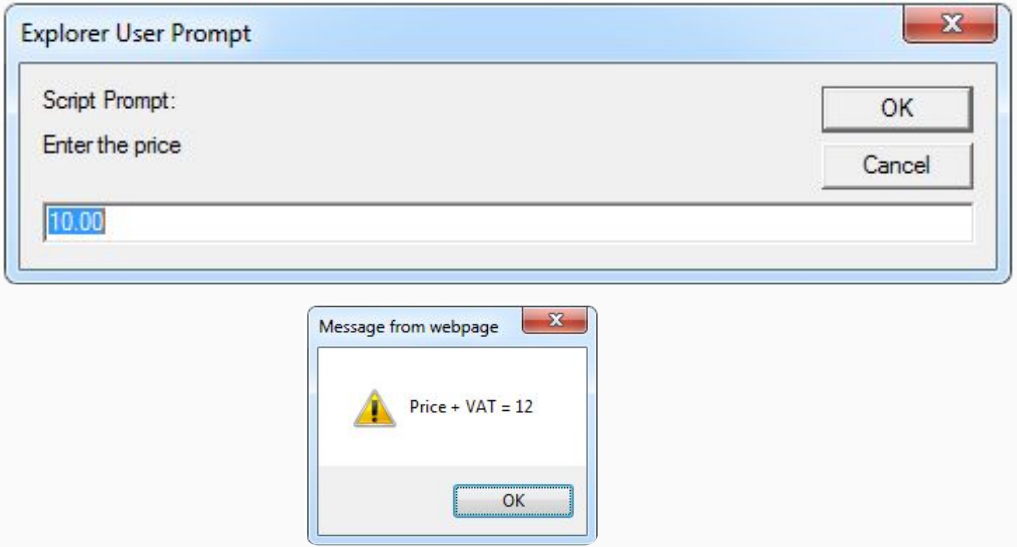
script.js

```
function calcSum() {
      value1 =
        parseInt(document.mainForm.textBox1.value);
      value2 =
        parseInt(document.mainForm.textBox2.value);
      sum = value1 + value2;
      document.mainForm.textBoxSum.value = sum;
    }
```



JavaScript Prompt - Example

```
price = prompt("Enter the price", "10.00");
alert('Price + VAT = ' + price * 1.24);
```



DOM

DOM

- The Document Object Model (DOM) is a standard object model and programming interface (API) for HTML.
- It defines:
  - The HTML elements as objects
  - The properties of all HTML elements
    - = a value that you can get or set
  - The methods to access all HTML elements
    - = an action that you can do
  - The events for all HTML elements

The DOM Programming Interface

- **The programming interface = the set of properties and methods.**
- **Document**
  - **If you want to access any element in an HTML page, always start with the document:**

```
// by id
document.getElementById("myId");

// by class name
document.getElementsByClassName("myClass");

// by tag name
document.getElementsByTagName("div");
```

DOM

- **Javascript gets all the power it needs to create dynamic HTML:**
  - JavaScript can change all the HTML elements in the page
  - JavaScript can change all the HTML attributes in the page
  - JavaScript can change all the CSS styles in the page
  - JavaScript can remove existing HTML elements and attributes
  - JavaScript can add new HTML elements and attributes
  - JavaScript can react to all existing HTML events in the page
  - JavaScript can create new HTML events in the page

Common DOM Events

- **Mouse events:**
  - `onclick`, `onmousedown`, `onmouseup`
  - `onmouseover`, `onmouseout`, `onmousemove`
- **Key events:**
  - `onkeypress`, `onkeydown`, `onkeyup`
  - **Only for input fields**
- **Interface events:**
  - `onblur`, `onfocus`
  - `onscroll`

Common DOM Events

- Miscellaneous events
  - `onload`, `onunload`
    - Allowed only for the `<body>` element
    - Fires when all content on the page was loaded / unloaded

onload Event - Example

- `onload` event

test.html

```
<html>

<head>
    <script
type="text/javascript"
src="script.js"> </script>
</head>

<body onload="greet()" >
</body>

</html>
```

Script.js

```
function greet() {
      alert("Loaded.");
    }
```

Working with DOM

Finding elements in DOM

```
// by id
document.getElementById("myId");


// by class name
document.getElementsByClassName("myClass");

// by tag name
document.getElementsByTagName("div");
```

Changing HTML elements (working with getters and setters)

```
// GET
// return the content for the selected element
document.getElementById("myId").innerHTML;


// SET
// updates the content for the selected element
document.getElementById("myId").innerHTML = "Some text";
```

Summary

Hello World! In JavaScript

- Create an HTML file
- Create a JS file and include it in that HTML file
- Add a function that alerts “Hello World!” on page load
- Add another function that displays “Hello World!” inside a div with id “main” on the page
- Add another function that displays “Hello World!” inside a div with id “after-click” after a button with id=”button” was clicked

Resources

CodeCademy JavaScript tutorial

MDN JavaScript Guide (until Keyed collections - including it)