

CENG420

# Chapter 4 JavaScript

Part 1



# What is it?

- o Developed by Netscape.
- o JavaScript: Done on the client side, by the browser, alternative for some of what's done with server-side programming.
- o Describes interactions with buttons, elements, and menus.

# Embedding JS in HTML

- o two ways (both in head section)

- o **Way 1:**

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

- o **Way 2:**

```
<script type="text/javascript">
```

```
//Your javascript code
```

```
</script>
```





# **1** JavaScript Basics

# Primitives

- Numbers, strings, boolean
- examples: 72, 7.2, .72, .7e2, 'hello', "hello"
- more examples: ' you\'re the most welcome'
- using the backslash is called character escaping: tells the browser to print a quote instead of considering it as closing the string.
- Can be used with special characters like slash
- Example: "D:\\books" will appear D:\books

# Declaring variables

- o No need to specify type in JS.

- o we can use the keyword **var**

```
a=7
```

```
(++a) * 3 = 24 and a = 8
```

```
(a++) * 3 = 21 and a = 8
```

```
var a = 2;
```

```
a= "hello"
```

- o Variable names can start with \$, \_ or a letter



# Math objects

- o have properties and methods for “number” objects: sin, cos, floor,...
- o `math.Sin(x)`; `MAX_VALUE`, ...
- o `isNaN(a)` returns true if a is not a number.
- o `toString()` converts numbers to strings
- o **Example**

```
var price=2.0, str_price;  
str_price = price.toString()
```

# Strings

- o Concatenation: the **+** sign
- o Example: “hello” + “ there” = “hello there”
- o “aug” + 1977 = “aug1977”
- o JS always consider variables as strings with the + sign.
- o This is called implicit conversion



# Strings (cont.)

- Explicit conversion:

```
var num=6;  
var str_val = num.toString(); // Decimal as 6, convert it to string "6"  
var str_val = num.toString(2) // Binary 6 (110) convert to string "110"
```

- We can convert string to number if the string is only a number “6” for example: 2 ways:
  - Way 1:** `var x = Number(mystring);`
  - Way 2:** `var x= mystring - 0;`
- We can parse an integer out of string using `parseInt`

# Strings

- o Methods to manipulate strings and properties like length `str.length`
- o Methods:
  - o `charAt(i)` returns the character at index `i`
  - o `indexOf('c')` returns index of character `c`.
  - o `substring(1,3)` returns string from index 1 to 3
  - o `toLowerCase()`

# String Object Methods

Method	Description
<a href="#"><u>charAt()</u></a>	Returns the character at the specified index
<a href="#"><u>charCodeAt()</u></a>	Returns the Unicode of the character at the specified index
<a href="#"><u>concat()</u></a>	Joins two or more strings, and returns a copy of the joined strings
<a href="#"><u>fromCharCode()</u></a>	Converts Unicode values to characters
<a href="#"><u>indexOf()</u></a>	Returns the position of the first found occurrence of a specified value in a string
<a href="#"><u>lastIndexOf()</u></a>	Returns the position of the last found occurrence of a specified value in a string
<a href="#"><u>localeCompare()</u></a>	Compares two strings in the current locale
<a href="#"><u>match()</u></a>	Searches for a match between a regular expression and a string, and returns the matches
<a href="#"><u>replace()</u></a>	Searches for a match between a substring (or regular expression) and a string, and replaces the matched substring with a new substring
<a href="#"><u>search()</u></a>	Searches for a match between a regular expression and a string, and returns the position of the match
<a href="#"><u>slice()</u></a>	Extracts a part of a string and returns a new string
<a href="#"><u>split()</u></a>	Splits a string into an array of substrings
<a href="#"><u>substr()</u></a>	Extracts the characters from a string, beginning at a specified start position, and through the specified number of character
<a href="#"><u>substring()</u></a>	Extracts the characters from a string, between two specified indices



# String methods (cont)

<u>toLocaleLowerCase()</u>	Converts a string to lowercase letters, according to the host's locale
<u>toLocaleUpperCase()</u>	Converts a string to uppercase letters, according to the host's locale
<u>toLowerCase()</u>	Converts a string to lowercase letters
<u>toString()</u>	Returns the value of a String object
<u>toUpperCase()</u>	Converts a string to uppercase letters
<u>trim()</u>	Removes whitespace from both ends of a string
<u>valueOf()</u>	Returns the primitive value of a String object

- We can also compare strings using < and >

# HTML document

- o html document → *document* object
- o Window displaying html → *window* object

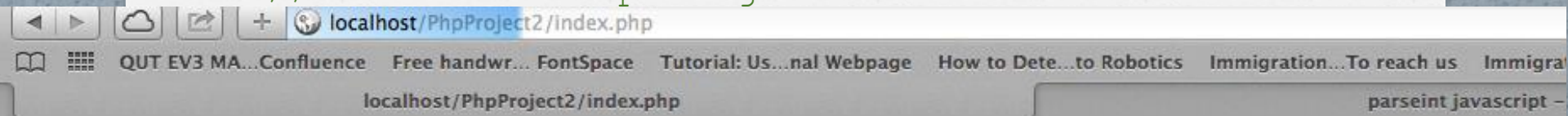
```
document.write ("hello" + x + "<br/>");
```

- o this will create html code, that's why we can either write a string, a variable and of course html code!!!

# Window object

`o` has: alert, confirm, prompt

```
alert("the sum is" + sum);  
// doesn't accept tags since this doesn't
```



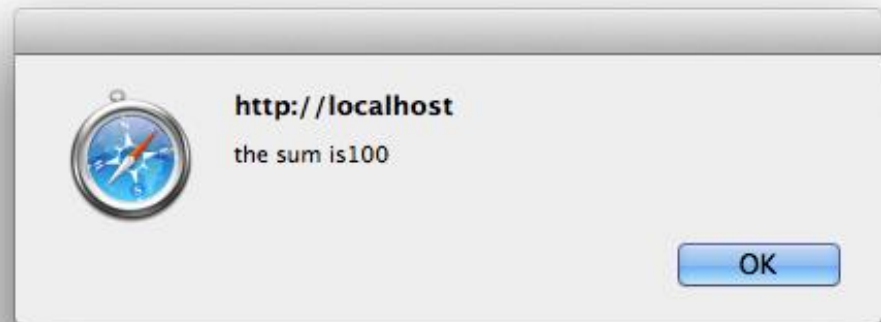
hello2  
1977aug

this is p by itself

this is p with class = ceng450

this is generic h1

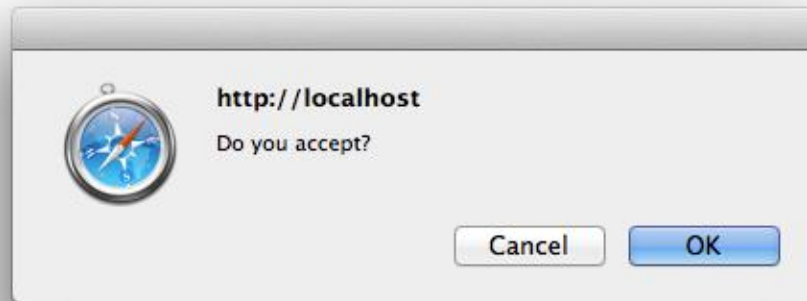
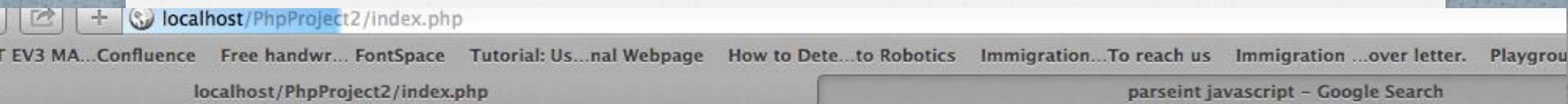
this is a generic p





# Window object (cont)

```
var question = confirm("Do you accept?");  
return true or false
```

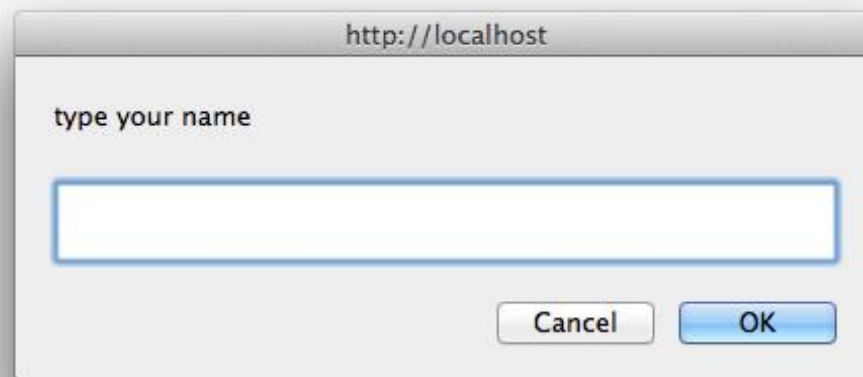


# Window object (cont)

```
name = prompt("type your name","default");
```

- default value is needed in case user didn't enter anything.

Confluence Free handwr... FontSpace Tutorial: Us...nal Webpage How to Det...to Robotics Immigration...To reach us Immigration ...over letter. Playground  
localhost/PhpProject2/index.php parseint javascript - Google Search



## ❑ Control

- if (condition) then else
- Switch

```
switch(n)
```

```
{
```

```
case 1:
```

```
    execute code block 1
```

```
    break;
```

```
case 2:
```

```
    execute code block 2
```

```
    break;
```

```
default:
```

```
    code to be executed if n is different from case 1 and 2
```

```
}
```



# loops

## Looping

- ❑ *for( init ; control ; inc)*  
for (var i=0;i<cars.length;i++)  
{  
document.write(cars[i] + "<br>");  
}
- ❑ *while ( control )*
- ❑ *do ... while*

# Control and Looping

---

## ❑ Control

- if (condition) then else
- switch

## ❑ Looping

- for( *init ; control ; inc* )
- while ( *control* )
- do ... while
- for .. in
  - for (*property in object*) {

# Arrays

- Are objects
- *dynamic* length
- Can be primitive or reference to objects such as other arrays

```
var myarray= new Array(1,2,"three");
```

```
var myarray= [1,2,"three"]
```



# Arrays

- o Suppose `myarray` was only 4 elements and you do this:

```
myarray[47] = 22;
```

- o Now the array is 48 cells! 😊

```
var x = myarray.length
```

```
myarray.length = 1000; // change length of array
```

- o Only assigned values occupy a space.

# Array methods

- o **join**: convert elements to string and concatenate them into a single string
  - o no params: they will be separated by commas `join()`
  - o param: separated by parameter. Example: `join('-')`
- o **reverse()**: reverse the order of elements
- o **sort()**: convert to string and sort alphabetically
- o **concat(values)**: adds values to end of array

# Array methods

o **slice** → `var list = [2,4,8,16]`

`var list2=list.slice(1,3)`

→ `list2 = [4,8]` elements 1 and 2 not including 3!

`list3 = list.slice(2)`

→ `[8,16]` element 2 and beyond

o

o **toString** : same as join

o **push, pop, unshift, shift**

o 2-D arrays: `n = [ [2,3,4], [1,3,5] ]`



## example

```
<html>
  <head>
    <script type="text/javascript">
      var e=5;
      e="hello";
      a = [1,2,"f"];
    </script>

    <title>TODO supply a title</title>
    <meta charset="UTF-8">
    <meta name="viewport"
      content="width=device-width, initial-scale=1.0">
  </head>
  <body>
    <p >hello class</p>
  </body>
</html>
```



# Functions

# Functions

- o `function fun() { //content }`
- o `ref_fun = fun` // assign ref\_fun to fun
- o `fun()` and `ref_fun ()` both now call the same function
- o Local variables inside a function have precedence over global ones (if they have the same name)



**example**

```
<script>  
    function showmessage(m)  
    {  
        alert(m) ;  
    }  
</script>
```



Who do you think will call this function?



# Pattern matching

# What is it and why do we need it?

- o Patterns are user defined series of characters and number.
- o for example, 3 letters followed by a number followed by a capital letter.
- o **Why do we need it?**
  - o Passwords for example are required to have a specific structure and length, just like your LIU password
  - o Other uses as well.



# Pattern matching

## Pattern Matching

- ❑ Patterns are specified as regular expressions
  - Based on Perl's regular expressions
- ❑ Two approaches
  - String object
    - ① `search( pattern )`
      - return start position of a match
      - returns -1 if no match
    - ② `replace( pattern, value )`
    - ③ `match( pattern )`
    - ④ `split( pattern )`
  - RegExp object

# Pattern matching: **search()**

- `\w`
  - An alphanumeric
- `\d`
  - A digit
- `\s`
  - A white space
- `.`
  - Anything other than newline
- `[abcde]`
  - Any of a,b,c,d,e
- `[a-e]`
  - Same as above
- `[^a-e]`
  - Anything but a to e
- `exp?`, `exp+`, `exp*`
  - 0 or 1, 1 or more, 0 or more
  - symbolic quantifiers
- `exp{x}`
  - Exactly x repeats
- `exp{x,y}`
  - At least x repeats, but no more than y repeats
- `expA | expB`
  - expA or expB
- `/exp/i`
  - Match either upper or lowercase in `exp`

## Strings and Regular Expressions

### ❑ match(regExpObj)

- Verify input

```
var phone = "416-4403467";  
if (phone.match(/^\d{3}-\d{7}/))  
    return true;
```

### ❑ replace(regExpObj, str)

- Replace part of a string

```
var str = "One elephant and two zebras";  
var matches = str.replace(/two/, "three");
```

The `match()` method searches a string for a match against a regular expression, and returns the matches, as an Array object.



# Examples on pattern matching

```
var str= "hello I am here"
```

```
var position = str.search(/lo/);
```

position -> 3 (returns first position)

◦ Meta characters: \ | ( ) [ ] { } ^ \$ \* + ? .

# More examples

- o `/snow ./` matches snowy, snowd,...
- o `/2\.4/` matches only 2.4 while `/2.4/` matches 2.4, 2a4, 294,....
- o character classes:
  - o `[abc]` matches 'a' or 'b' or 'c'
  - o `[a-h]` matches any character from a to h
  - o `[^aeiou]` matches any character except a,e,i,o,u,

# Predefined values

- o `\d` is equivalent to `[0-9]` matches one digit
- o `\D` is equivalent to `[^0-9]` matches anything but a digit.
- o `\w` is equivalent to `[A-Za-z_0-9]` matches a word character
- o `\W` = `[^A-Za-z_0-9]` anything but a word character
- o `\s` is white space



# More examples

- o `/\d\.\d\d/` matches a digit followed by dot followed by 2 digits → 3.25
- o `/\D\d\D/` matches a single digit like a5f
- o `/\w\w\w/` matches 3 adjacent characters like abc

# More examples

o `/xy{4}z/` matches



o `{i}` matches *i* repetitions while `{i,j}` matches at least *i* and up to *j* repetitions

o `/x*y+z?/` matches



or



o `/\d+\.\d*/`



# More examples

- o `/[A-Za-z]\w*/` letter followed by zero or more letters, digits or underscore
- o `\b` is boundary between `\w` and `\W`
- o `/\bis\b/` matches “he is good” but not “he isn’t”
- o  **Anchors:**
- o `/^pearl/` matches “pearls are” not “my pearls” (beginning)
- o `/pearl$/` matches “I like pearl” not “pearl is” (end)
- o Anchors should be first or last letter inside the pattern to mean anchors.



# Str= “Fred, Freddie and Frederica were siblings”

- o `str.replace(/Fre/g, "Boy")` where /g means global = replace all patterns
- o str becomes :Boyd, Boydie and Boyderica...
- o The matched substrings are automatically saved to 3 variables \$1, \$2 and \$3, all of them are set to “Fre” in this case

“having 4 apples is better than  
having 3 oranges”

```
matches = str.match(/\d/g)
```

o matches = [4,3]

I have 428 dollars, but I need  
500

```
matches = str.match(/(\d+) ([^\d]+) (\d+)/)
```

o = ["428 dollars, but I need 500", "428", "dollars, but I need", "500"]

o when () are used around expressions, first element of array is answer then each () answer by itself.



str="grapes:apples:oranges"

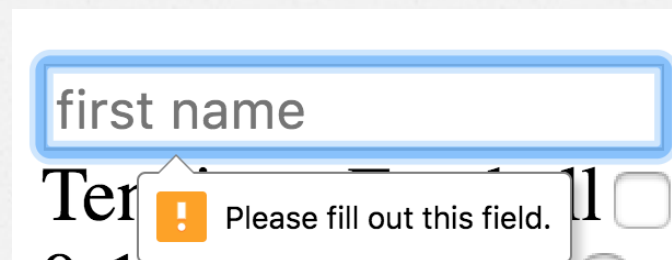
```
fruit = str.split(":");
```

```
o = [grapes, apples, oranges]
```

# Remember in html the required attribute

- When used inside `<input>` element, whenever the form is submitted, if this input is empty, we get a warning and the form is not submitted.

```
<input  
type='text'  
name='fname'  
placeholder='first name'  
required />
```



first name

Ter... 11

! Please fill out this field.


# the HTML **pattern** attribute

- in html5, we can use pattern attribute inside our element and when the form is submitted, the pattern is checked. Similar to the **required** attribute behavior but this one is for patterns.

```
<form action="/action_page.php">  
Country code: <input type="text" name="country_code"  
pattern="[A-Za-z]{3}" title="Three letter country code">  
<input type="submit">  
</form>
```

Country code:

No  
ver

 Please match the requested format.  
Three letter country code

ot



# Exercises

- 1) Prompt the user to enter values a and b and calculate  $a^2/b^2$  and display this in the page inside an `<h1>` tag.

# Exercises (cont.)

- 2) create a `<select>` list using `document.write` containing the countries in the array

```
var countries  
=['lebanon', 'uae', 'usa'];
```

# Exercises

- 3) Create a html input, let the user input a value. And click a button. If the value  $m$  is between 1 and 10, print a table that has  $m \times m$  size, each cell containing the index of the cell. If the value is not between 1 and 10, alert the user.
- Hints (some hints are from the next part of the lecture)**  
create a function **draw()** in js that reads the value from the input field and use **document.write** to produce the table.  
To read the value use:  
**`v = document.getElementById("idofinput").value`**
- To call the function when the button is clicked, add the attribute **onclick="draw()"** to the **button** html



# Advice

- You can install a plugin called *JSLint* or *JSHint* inside netbeans to check the syntax errors of the JS code
- Tools → plugins



2

# JavaScript Events

more in chapter 5

# What are they?

- o JavaScript events
  - allow scripts to respond to user interactions and modify the page accordingly
- o Events and event handling
  - help make web applications more dynamic and interactive



# Load event

- o The window object's Load event fires when the window finishes loading successfully (i.e., all its children are loaded and all external files referenced by the page are loaded)
- o An **event handler** is a function that responds to an event.

# Load event

- ▶ **Two** models for registering event handlers
  - **Inline** model treats events as attributes of HTML elements
  - **Traditional** model assigns the name of the function to the event property of a DOM node
- ▶ The inline model places calls to JavaScript functions directly in HTML code.
- ▶ The following code indicates that JavaScript function **start** should be called when the **body** element loads:

```
<body onload = "start()">
```

# Load event

- ▶ The traditional model uses a property of an object to specify an event handler.
- ▶ The following JavaScript code indicates that function start should be called when document loads:

```
document.onload = "start()";
```



# Accessing an element in the page from JS

```
document.getElementById(id_of_the_element)
```

# HTML DOM **events**

## Mouse Events

Property	Description
<u><a href="#">onclick</a></u>	The event occurs when the user clicks on an element
<u><a href="#">ondblclick</a></u>	The event occurs when the user double-clicks on an element
<u><a href="#">onmousedown</a></u>	The event occurs when a user presses a mouse button over an element
<u><a href="#">onmousemove</a></u>	The event occurs when the pointer is moving while it is over an element
<u><a href="#">onmouseover</a></u>	The event occurs when the pointer is moved onto an element
<u><a href="#">onmouseout</a></u>	The event occurs when a user moves the mouse pointer out of an element
<u><a href="#">onmouseup</a></u>	The event occurs when a user releases a mouse button over an element

## Keyboard Events

Attribute	Description
<u><a href="#">onkeydown</a></u>	The event occurs when the user is pressing a key
<u><a href="#">onkeypress</a></u>	The event occurs when the user presses a key
<u><a href="#">onkeyup</a></u>	The event occurs when the user releases a key

# Frame/Object Events

Attribute	Description
onabort	The event occurs when an image is stopped from loading before completely loaded (for <object>)
onerror	The event occurs when an image does not load properly (for <object>, <body> and <frameset>)
<u>onload</u>	The event occurs when a document, frameset, or <object> has been loaded
<u>onresize</u>	The event occurs when a document view is resized
onscroll	The event occurs when a document view is scrolled
<u>onunload</u>	The event occurs once a page has unloaded (for <body> and <frameset>)

# Form Events

Attribute	Description
<u>onblur</u>	The event occurs when a form element loses focus
<u>onchange</u>	The event occurs when the content of a form element, the selection, or the checked state have changed (for <input>, <select>, and <textarea>)
<u>onfocus</u>	The event occurs when an element gets focus (for <label>, <input>, <select>, textarea>, and <button>)
onreset	The event occurs when a form is reset
<u>onselect</u>	The event occurs when a user selects some text (for <input> and <textarea>)
onsubmit	The event occurs when a form is submitted



# EventTarget

## EventTarget Object

### Methods

Method	Description
<code>addEventListener()</code>	Allows the registration of event listeners on the event target (IE8 = <code>attachEvent()</code> )
<code>dispatchEvent()</code>	Allows to send the event to the subscribed event listeners (IE8 = <code>fireEvent()</code> )
<code>removeEventListener()</code>	Allows the removal of event listeners on the event target (IE8 = <code>detachEvent()</code> )

# MouseEvent/KeyboardEvent Object

## Properties

Property	Description
<u>altKey</u>	Returns whether or not the "ALT" key was pressed when an event was triggered
<u>button</u>	Returns which mouse button was clicked when an event was triggered
<u>clientX</u>	Returns the horizontal coordinate of the mouse pointer, relative to the current window, when an event was triggered
<u>clientY</u>	Returns the vertical coordinate of the mouse pointer, relative to the current window, when an event was triggered
<u>ctrlKey</u>	Returns whether or not the "CTRL" key was pressed when an event was triggered
keyIdentifier	Returns the identifier of a key
keyLocation	Returns the location of the key on the device
<u>metaKey</u>	Returns whether or not the "meta" key was pressed when an event was triggered
<u>relatedTarget</u>	Returns the element related to the element that triggered the event
<u>screenX</u>	Returns the horizontal coordinate of the mouse pointer, relative to the screen, when an event was triggered
<u>screenY</u>	Returns the vertical coordinate of the mouse pointer, relative to the screen, when an event was triggered
<u>shiftKey</u>	Returns whether or not the "SHIFT" key was pressed when an event was triggered