CENG420

Chapter 4 JavaScript

Part 1

What is it?

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

Embedding JS in HTML

- two ways (both in head section)

- **Way 1:**

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

- **Way 2:**

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


//Your javascript code


</script>
```

❶ 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

- **No need** to specify type in JS.
- 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"
```

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

Math objects

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

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

Strings

- Concatenation: the **+ sign**
- Example: “hello” + “ there” = “hello there”

- “aug” + 1977 = “aug1977”
- JS always consider variables as strings with the + sign.
- 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

- Methods to manipulate strings and properties like length `str.length`

- Methods:
  - **`charAt(i)`** returns the character at index I
  - **`indexOf('c')`** returns index of character c.
  - **`substring(1,3)`** returns string from index 1 to 3
  - **`toLowerCase()`**

String Object Methods

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

String methods (cont)

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

- We can also compare strings using < and >

HTML document

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

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

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

Window object

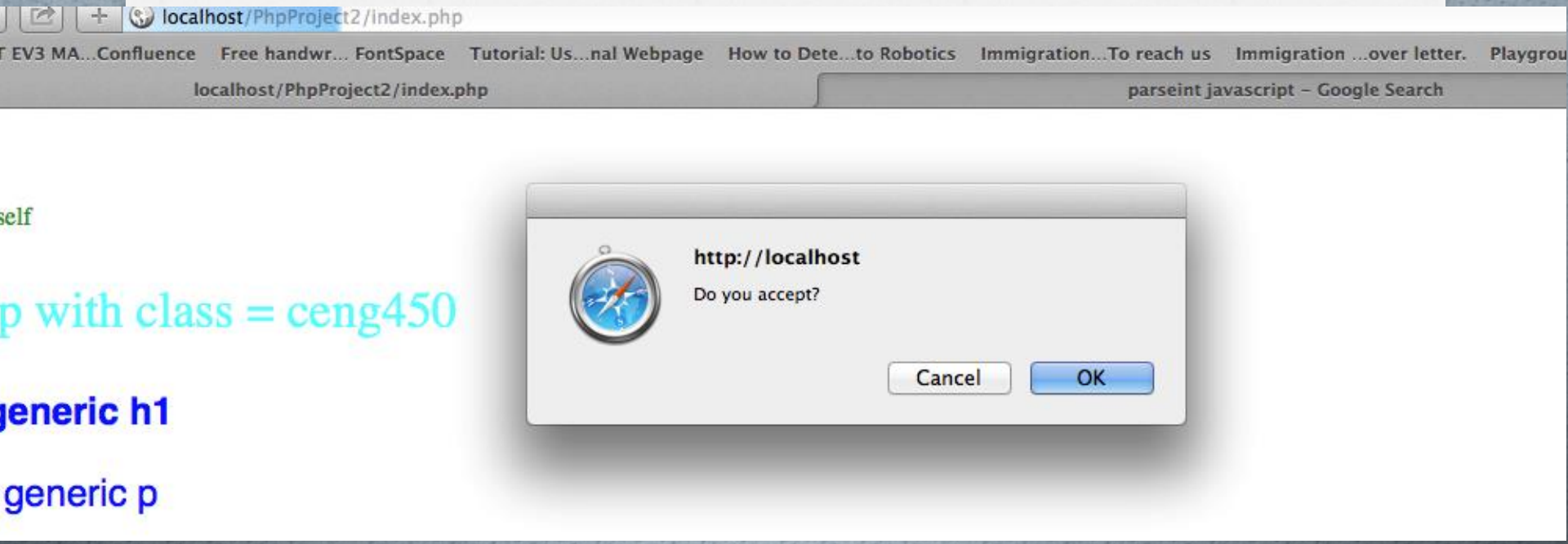
- has: `alert, confirm, prompt`

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

Window object (cont)

```
var question = confirm("Do you accept?");
```

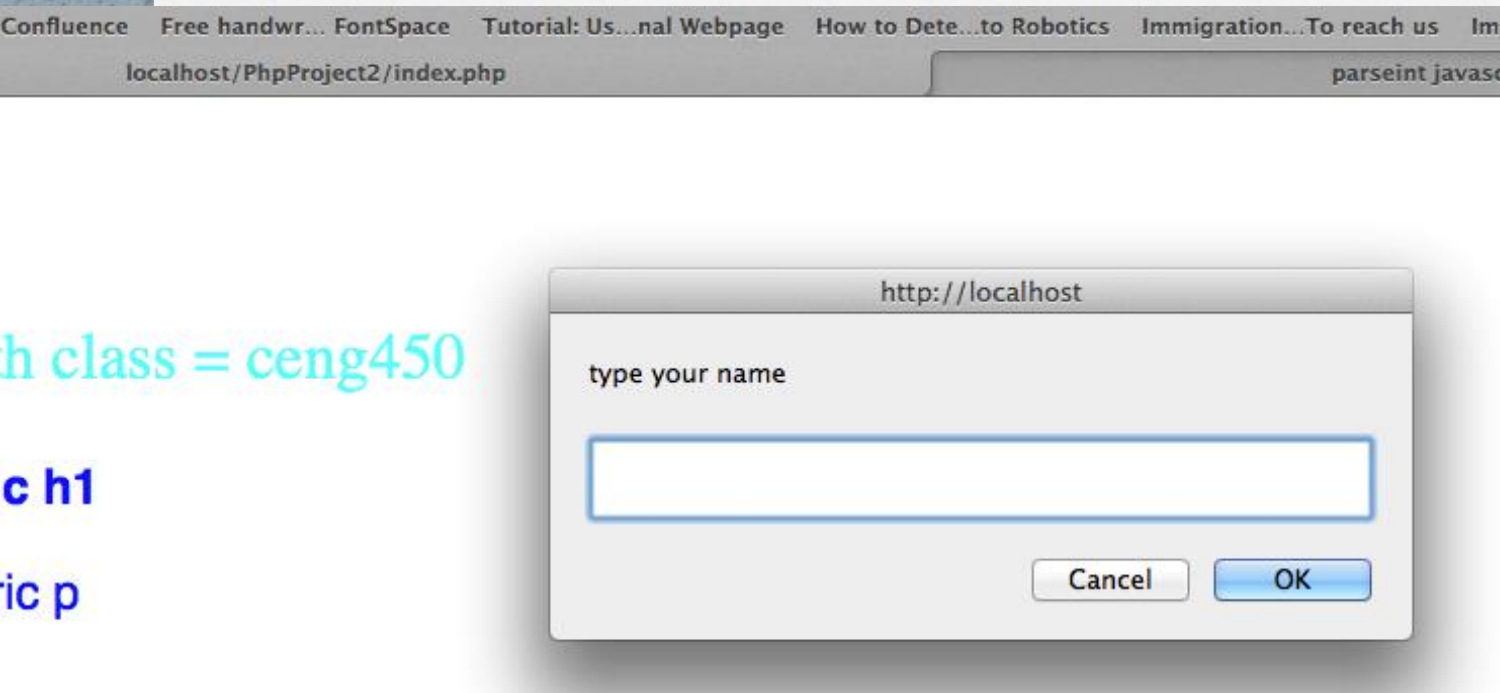
- return true or false



Window object (cont)

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

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



- 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

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

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

- Now the array is 48 cells! ☺

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

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

- Only assigned values occupy a space.

Array methods

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

Array methods

- **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

-
- **toString** : same as join
- **push, pop, unshift, shift**

- 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

- `function fun() { //content}`
- `ref_fun = fun` // assign ref_fun to fun
- fun() and ref_fun () both now call the same function

- Local variables inside a function have precedence over global ones (if they have the same name)

example

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

i Who do you think will call this function?

Pattern matching

What is it and why do we need it?

- Patterns are user defined series of characters and number.
- for example, 3 letters followed by a number followed by a capital letter.

- Why do we need it?
  - Passwords for example are required to have a specific structure and length, just like your LIU password
  - 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

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

Predefined values

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

More examples

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

More examples

- /xy{4}z/ matches [redacted]
- {i} matches i repetitions while {i,j} matches at least i and up to j repetitions
- /x*y+z?/ matches [redacted] or [redacted]
- /\d+\.\d*/ [redacted]

More examples

- **`/[A-Za-z]\w*/`** letter followed by zero or more letters, digits or underscore

- **`\b`** is boundary between \w and \W
- **`/\bis\b/`** matches “he is good” but not “he isn’t”

- Anchors:
- **`/^pearl/`** matches “pearls are” not “my pearls” (beginning)
- **`/pearl$/`** matches “I like pearl” not “pearl is” (end)
- Anchors should be first or last letter inside the pattern to mean anchors.

Str= “Fred, Freddie and Frederica were siblings”

- **`str.replace(/Fre/g, “Boy”)`** where /g means global = replace all patterns

- str becomes :Boyd, Boydie and Boyderica…

- 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)
```

- matches = [4,3]

I have 428 dollars, but I need 500

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

- = [“428 dollars, but I need 500”, “428”, “dollars, but I need”, “500”]

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

str=“grapes:apples:oranges”

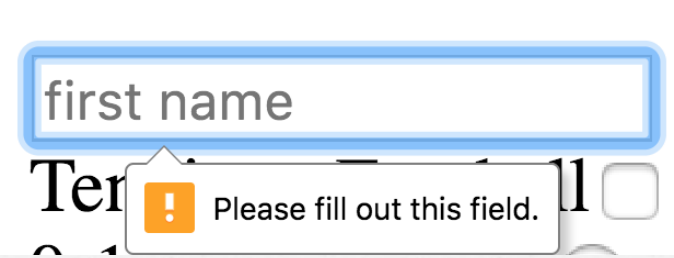
```
fruit = str.split(“:”);
```

- = [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.

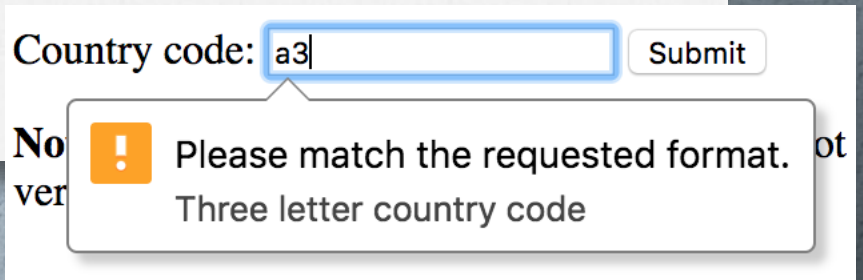
```
<input
type='text'
name='fname'
placeholder='first name'
required />
```



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>
```



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 x 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 form 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

❷ JavaScript Events

more in chapter 5

What are they?

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

Load event

- 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)

- 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
onclick	The event occurs when the user clicks on an element
ondblclick	The event occurs when the user double-clicks on an element
onmousedown	The event occurs when a user presses a mouse button over an element
onmousemove	The event occurs when the pointer is moving while it is over an element
onmouseover	The event occurs when the pointer is moved onto an element
onmouseout	The event occurs when a user moves the mouse pointer out of an element
onmouseup	The event occurs when a user releases a mouse button over an element

Keyboard Events

Attribute	Description
onkeydown	The event occurs when the user is pressing a key
onkeypress	The event occurs when the user presses a key
onkeyup	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>)
onload	The event occurs when a document, frameset, or <object> has been loaded
onresize	The event occurs when a document view is resized
onscroll	The event occurs when a document view is scrolled
onunload	The event occurs once a page has unloaded (for <body> and <frameset>)

Form Events

Attribute	Description
onblur	The event occurs when a form element loses focus
onchange	The event occurs when the content of a form element, the selection, or the checked state have changed (for <input>, <select>, and <textarea>)
onfocus	The event occurs when an element gets focus (for <label>, <input>, <select>, textarea>, and <button>)
onreset	The event occurs when a form is reset
onselect	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
addEventListener()	Allows the registration of event listeners on the event target (IE8 = attachEvent())
dispatchEvent()	Allows to send the event to the subscribed event listeners (IE8 = fireEvent())
removeEventListener()	Allows the removal of event listeners on the event target (IE8 = detachEvent())

MouseEvent/KeyboardEvent Object

Properties

Property	Description
altKey	Returns whether or not the "ALT" key was pressed when an event was triggered
button	Returns which mouse button was clicked when an event was triggered
clientX	Returns the horizontal coordinate of the mouse pointer, relative to the current window, when an event was triggered
clientY	Returns the vertical coordinate of the mouse pointer, relative to the current window, when an event was triggered
ctrlKey	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
metaKey	Returns whether or not the "meta" key was pressed when an event was triggered
relatedTarget	Returns the element related to the element that triggered the event
screenX	Returns the horizontal coordinate of the mouse pointer, relative to the screen, when an event was triggered
screenY	Returns the vertical coordinate of the mouse pointer, relative to the screen, when an event was triggered
shiftKey	Returns whether or not the "SHIFT" key was pressed when an event was triggered

CENG420

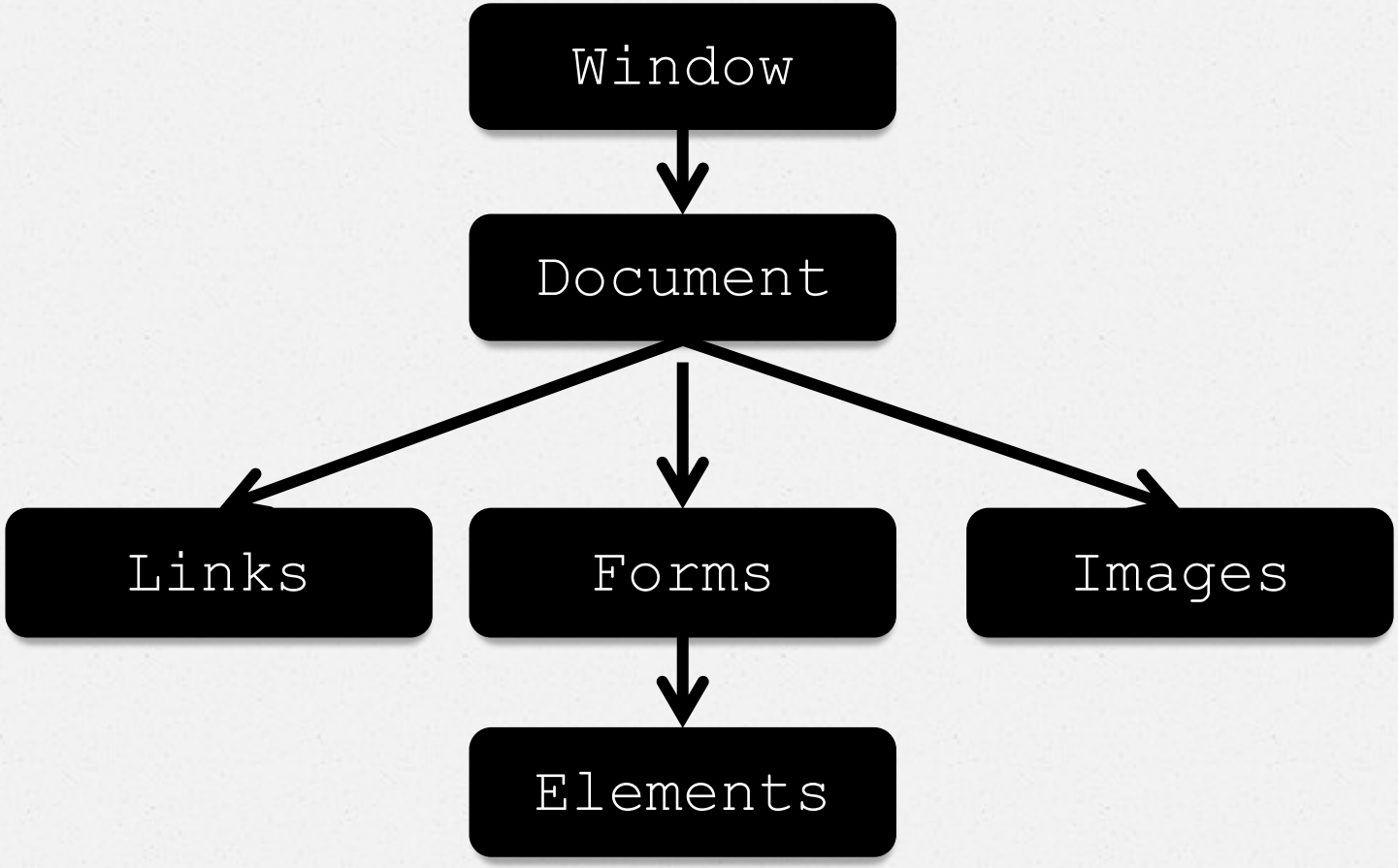
Chapter 4 JavaScript

Part 2

Hierarchy

- At the top level of the hierarchy, we have the window object
- Under the window object, we have the Document object.
- Inside the document object, we have Forms array (along with other arrays like anchors, links, images, …)
- Inside forms array, we have elements property
- All this is used to access document elements from Javascript.

Hierarchy (2)



DOM

Document Object Model

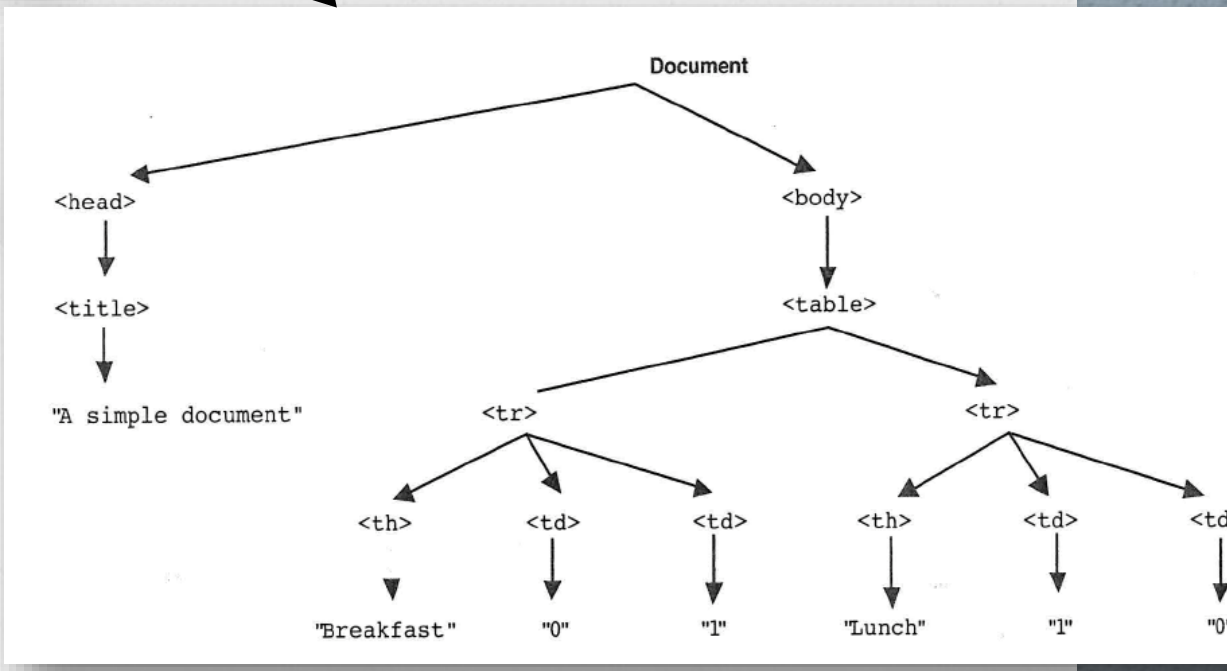
DOM
Document Object Model

- DOM0: used by early browsers that supported javascript

- DOM: API (Application Programming Interface) defines the interface between XHTML documents and application programs

Example

```
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head> <title> A simple document </title>
  </head>
  <body>
    <table>
      <tr>
        <th> Breakfast </th>
        <td> 0 </td>
        <td> 1 </td>
      </tr>
      <tr>
        <th> Lunch </th>
        <td> 1 </td>
        <td> 0 </td>
      </tr>
    </table>
  </body>
</html>
```



5.3 Element Access

Consider the following

```
<head> <title> Access to form elements </title>
</head>
<body>
  <form action = "">
    <input type = "button"  name = "turnItOn" />
  </form>
</body>
```

Question 1: how to access the form and the inputs inside that form from javascript?

Question 2: Why would we need this?

Solution 1

- Use the DOM forms array and elements array:

```
var dom = document.forms[0].elements[0];
```

- This will access the first `element` of the first `form` of the document → what is inside the variable `dom` now?
- What's the problem?

Solution 2

- Use “names”:

```
<form name = "myForm"  action = "">
  <input type = "button"  name = "turnItOn" />
</form>
```

```
var dom = document.myForm.turnItOn;
```

- Drawbacks: HTML 1.1 doesn’t support form names. This is not really a drawback since newer versions support it.
- Names are still used and (should be) with elements, especially when php is involved.

Solution 3

- Use “id” and getElementById()

```
<form name = "myForm"  action = "">
  <input type = "button"   id   = "turnItOn" />
</form>
```

```
var dom = document.getElementById("turnItOn");
```

- IDs are unique and they can be used safely no matter how deep is the element in the document.
- **What happens when we have checkboxes or radio?**

Accessing radio/checkboxes

- A mix of names and ids is used for this case!

```
<form id = "vehicleGroup">
  <input type = "checkbox"  name = "vehicles"
         value = "car" />  Car
  <input type = "checkbox"  name = "vehicles"
         value = "truck" />  Truck
  <input type = "checkbox"  name = "vehicles"
         value = "bike" />  Bike
</form>
```

- dom holds the form
- Dom.vehicles is an array
- Dom.vehicles[i] is element i of that array

```
var numChecked = 0;
var dom = document.getElementById("vehicleGroup");
for (index = 0; index < dom.vehicles.length; index++)
  if (dom.vehicles[index].checked)
    numChecked++;
```

Other useful functions

document.getElementById()	Returns the element that has the ID attribute with the specified value
document.getElementsByClassName()	Returns a NodeList containing all elements with the specified class name
document.getElementsByName()	Returns a NodeList containing all elements with a specified name
document.getElementsByTagName()	Returns a NodeList containing all elements with the specified tag name

Manipulating HTML Elements

To access an HTML element from JavaScript,

- you can use the **`document.getElementById(id)`** method.
- Use the ***id* attribute** to identify the HTML element
- and **`innerHTML`** to refer to the element ***content***

```
<!DOCTYPE html>
<html>
<body>
<h1>My First Web Page</h1>
<p id="demo">My First Paragraph.</p>
<script>
document.getElementById("demo").innerHTML = "Paragraph changed.";
</script>
</body>
</html>
```

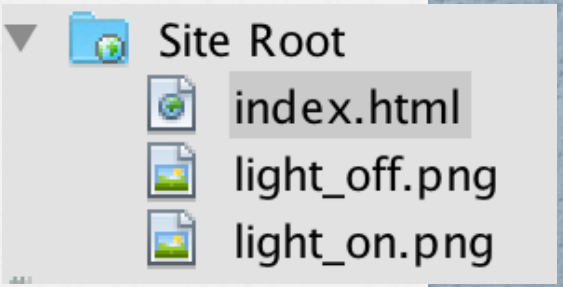
JavaScript Can Change HTML Elements

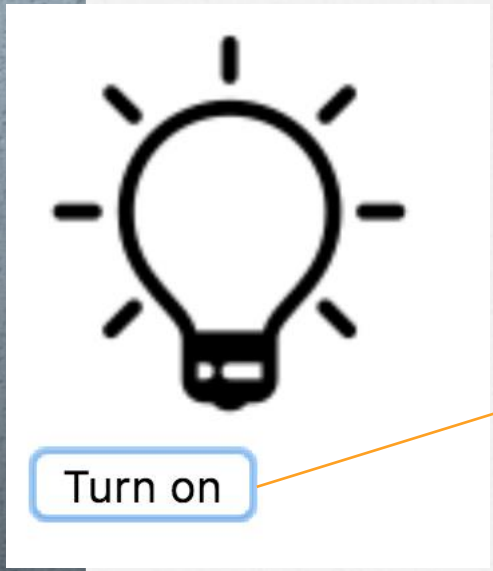
```
<!DOCTYPE html>
<html>
<body>

<h1>My First JavaScript</h1>
<p>JavaScript can change the content of an HTML
   element:</p>

<button type="button"
   onclick="myFunction()">Click Me!</button>

<p id="demo">This is a demonstration.</p>
<script>
function myFunction() {
    document.getElementById("demo").innerHTML =
   "Hello JavaScript!";
}
</script>
</body>
</html>
```

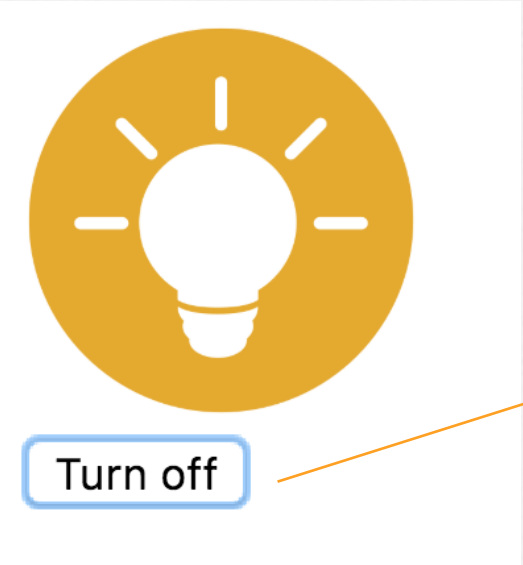
Accessing attributes

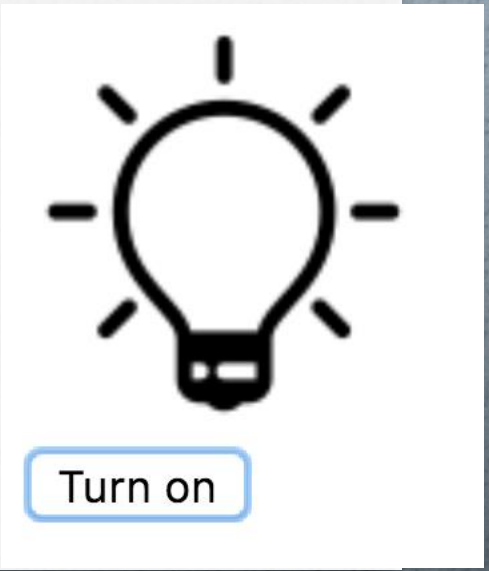
Changing HTML **Attributes**

- Assume you have two images in your folder called light_on and light_off.
- Originally, the off is shown and there's a button saying turn on, when clicked, the image becomes on, and the button says turn off









Changing HTML **Attributes (cont.)**

HTML part

```
<body>
    <img id='light'
    src='light_off.png'
    alt='light'
    width="100"
    height='100'/>

    <br/>

    <button id='lightswitch'
    onclick="toggle()"> Turn on
    </button>
</body>
```

Changing HTML **Attributes (cont.)**

JS part

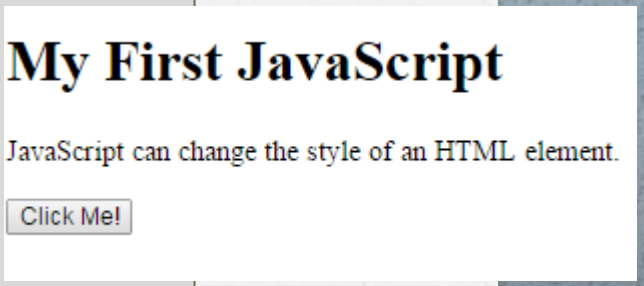
```
function toggle()
{
    image_element = document.getElementById('light');
    button_element = document.getElementById('lightswitch');

    if((image_element.src).match('light_off'))
        { image_element.src='light_on.png';
          button_element.innerHTML = "Turn off"; }

    else
        { image_element.src='light_off.png';
          button_element.innerHTML = "Turn on"; }
}
```

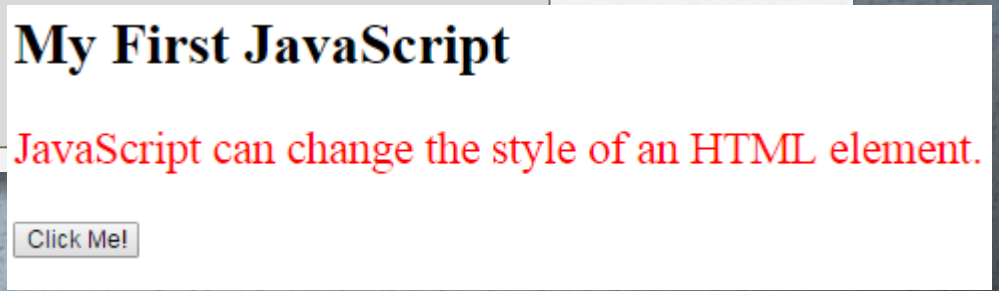
Notice how we accessed the `.src` attribute

Accessing style

JavaScript Can Change HTML Styles (CSS)

```
<h1>My First JavaScript</h1>

<p id="demo">JavaScript can change the style of an
   HTML element.</p>

<script>
function myFunction() {
    var x = document.getElementById("demo");
    x.style.fontSize = "25px";
    x.style.color = "red";
}
</script>

<button type="button" onclick="myFunction()">
   Click Me!</button>
```





we can also access CSS classes from JS

- Add, remove and change classes from JS

- **Method 1:**
- use the `**className**` **property in JS**

- **Assume you have 2 CSS classes myclass1 and myclass2.**

we can also access CSS classes from JS

- Assume you have 2 CSS classes myclass1 and myclass2.
- To assign a class:

```
X = document.getElementById('mydiv');
X.className='myclass1';
```

To add a class: `X.className +='myclass2';`

To remove classes: `X.className='';`

the `classList` property

- **Method 2:** use the property classList

```
document.getElementById("myDIV").classList.add("mystyle"
, "anotherClass", "thirdClass");
```

```
document.getElementById("myDIV").classList.remove("mysty
le");
```

JavaScript Can Validate Data

```
<p>Please input a number between 1 and 10:</p>
<input id="numb" type="number">
<button type="button" onclick="myFunction()">Submit</button>
<p id="demo"></p>

<script>
function myFunction() {
    var x, text;
    // Get the value of input field with id="numb"
    x = document.getElementById("numb").value;

    // If x is Not a Number or less than 1 or greater than 10
    if (isNaN(x) || x < 1 || x > 10) {
        text = "Input not valid";
    } else {
        text = "Input OK";}
    document.getElementById("demo").innerHTML = text;}
</script>
```

Please input a number between 1 and 10:

Submit

Creating elements

- You can create element in JS not just access them.
- Look at example next slide.

Example

- Assume we want to add a <select> inside with the following options

```
<select name="drop1" id="Select1">
  <option value="0">Volvo</option>
  <option value="1">Saab</option>
  <option value="2">Mercedes</option>
  <option value="3">Audi</option>
</select>
```

solution

```
var myDiv = document.getElementById("myDiv");

//Create array of options to be added
var array = ["Volvo","Saab","Mercedes","Audi"];

//Create and append select list
var selectList = document.createElement("select");
selectList.id = "mySelect";
myDiv.appendChild(selectList);

//Create and append the options
for (var i = 0; i < array.length; i++) {
    var option = document.createElement("option");
    option.value = i;
    option.text = array[i]; // we could have used innerHTML
    selectList.appendChild(option);
}
```

Example 2: Reading what the user has chosen in a <select>

Assume the user chose *volvo*.

```
<select id="myselect">
  <option value="0">Volvo</option>
  <option value="1">Saab</option>
  <option value="2">Mercedes</option>
  <option value="3">Audi</option>
</select>
```

Running this code:

```
var e = document.getElementById("myselect");
var strUser = e.options[e.selectedIndex].value;
```

Would make `strUser` be `0`.

If what you actually want is `volvo`, then do this:

```
var e = document.getElementById("myselect");
var strUser = e.options[e.selectedIndex].text;
```

5.4 Events &

Event handling

Events

- An event can happen to any element in the document
- Events like clicking, blurring, focus, loading, …
- check the next tables for a list of events.

List of events (1)

Event	Tag Attribute
blur	onblur
change	onchange
click	onclick
dblclick	ondblclick
focus	onfocus
keydown	onkeydown
keypress	onkeypress
keyup	onkeyup
load	onload

List of events (2)

Event	Tag Attribute
mousedown	onmousedown
mousemove	onmousemove
mouseout	onmouseout
mouseover	onmouseover
mouseup	onmouseup
reset	onreset
select	onselect
submit	onsubmit
unload	onunload

Note

- Same events may occur to several tags
- Not all tags have all events
- Check the tables.

Onblur and onchange

Attribute	Tag	Description
onblur	<a>	The link loses the input focus
	<button>	The button loses the input focus
	<input>	The input element loses the input focus
	<textarea>	The text area loses the input focus
	<select>	The selection element loses the input focus
onchange	<input>	The input element is changed and loses the input focus
	<textarea>	The text area is changed and loses the input focus
	<select>	The selection element is changed and loses the input focus

Other events

onclick	<a>	The user clicks on the link
	<input>	The input element is clicked
ondblclick	Most elements	The user double clicks the left mouse button
onfocus	<a>	The link acquires the input focus
	<input>	The input element receives the input focus
	<textarea>	A text area receives the input focus
	<select>	A selection element receives the input focus
onkeydown	<body>, form elements	A key is pressed down
onkeypress	<body>, form elements	A key is pressed down and released
onkeyup	<body>, form elements	A key is released
onload	<body>	The document is finished loading

Attribute	Tag	Description
onmousedown	Most elements	The user clicks the left mouse button
onmousemove	Most elements	The user moves the mouse cursor within the element
onmouseout	Most elements	The mouse cursor is moved away from being over the element
onmouseover	Most elements	The mouse cursor is moved over the element
onmouseup	Most elements	The left mouse button is unclicked
onreset	<form>	The reset button is clicked
onselect	<input>	The mouse cursor is moved over the element
	<textarea>	The text area is selected within the text area
onsubmit	<form>	The *Submit* button is pressed
onunload	<body>	The user exits the document

Examples

1

```
<input type = "button" id = "myButton"
       onclick = "alert('You clicked my button!');" />
```

```
<input type = "button" id = "myButton"
       onclick = "myButtonHandler();" />
```

2

```
document.getElementById("myButton").onclick =
                                    myButtonHandler;
```

5.5/6

Handling events from Body elements

Show a greeting when page loads

```
<body onload="load_greeting();">
  <p />
</body>
```

```
// load.js
//   An example to illustrate the load event

// The onload event handler
function load_greeting () {
  alert("You are visiting the home page of \n" +
        "Pete's Pickled Peppers \n" + "WELCOME!!!");
}
```

Handling events from elements

- Question: Alert the user whenever he clicks on a radio button about his choice

- Methodology: use onclick in each radio to call a function, let the function handle the alert.

We could have done this from the beginning

```
<form action="">


    Radio 1<input type="radio" name="r"  value="Radio1" onclick="choice(this)"/>
    Radio 2<input type="radio" name="r" value="Radio2" onclick="choice(this)"/>

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


    function choice(e){

        alert(e.value);
    }

</script>
```

- What do you think ***this*** represent?
- Notice how it's passed as an argument to the function

5.7 Examples

Example 1

- Design an online order form containing several items
- Next to each item, we have a field for the user to input the quantity
- A button at the bottom to click, shows the total cost
- When a user clicks on the total cost field, the field should blur (not allowing the user to type/change)

Item	Price	Qty
Cheese	1.00	
Pepperoni	2.00	
Pepper	1.00	
Salami	1.50	
TOTAL		

Item	Price	Qty
Cheese	1.00	4
Pepperoni	2.00	6
Pepper	1.00	1
Salami	1.50	2
TOTAL		20

```
<form action="">

            <p>Enter the desired quantities</p>
            <table border="2">

                <tr>
                    <th>Ingredient</th>
                    <th>Price</th>
                    <th>Quantity</th></tr>

                <tr>
                    <td>Extra Cheese</td>
                    <td>1.00</td>
                    <td><input type="text" id="xcheese" size="2" /></td>
                </tr><tr>
                    <td>Peperroni</td>
                    <td>0.75</td>
                    <td><input type="text" id="roni" size="2" /></td>
                </tr>
<tr>
                    <td>Green Pepper</td>
                    <td>0.50</td>
                    <td><input type="text" id="pep" size="2" /></td>
                </tr>
                <tr>
                    <td>Salami</td>
                    <td>2.00</td>
                    <td><input type="text" id="sal" size="2" /></td>
                </tr>
```

Example 1 (Cont)

```
<tr><td colspan="2">
<input type="button"  value="total Cost"
onclick="computetotal();" /></td>

<td> <input type="text" size="5" id="totalcost"
onfocus="this.blur();" /> </td></tr>
</table><p>
<input type="submit" value="Submit order" />

<input type="reset" value="Clear order form" />
            </p>
</form>
```

- Note the use of `this`
- `This` refers to the element it's found in.

JS part: the computeTotal function

```
function computetotal()
{
    var cheese_q = document.getElementById("xcheese").value;
    var roni_q   = document.getElementById("roni").value;
    var pepper_q = document.getElementById("pep").value;
    var salami_q = document.getElementById("sal").value;

    // the above could be replaced with a for loop instead of listing
them one
    // by one. can you figure out how? Changes should be made to both
html and js files


    var total    = cheese_q*1 + roni_q*0.75 + pepper_q*0.5 +salami_q*2;
    if (isNaN(total))
        document.getElementById("totalcost").value = "Error";
    else
        document.getElementById("totalcost").value = total;}
```

Example 2

- Validate input form user:
- let him enter 2 passwords
- While typing the first password, check if password is strong.
- After entering second, check if they match
- Let him enter a phone number, and check the format of that number
- Also a name, and you check the name format

Please input your details below

First name: milad

Password: ••• Password too short

Re-enter Password:

Phone:

submit form

Please input your details below

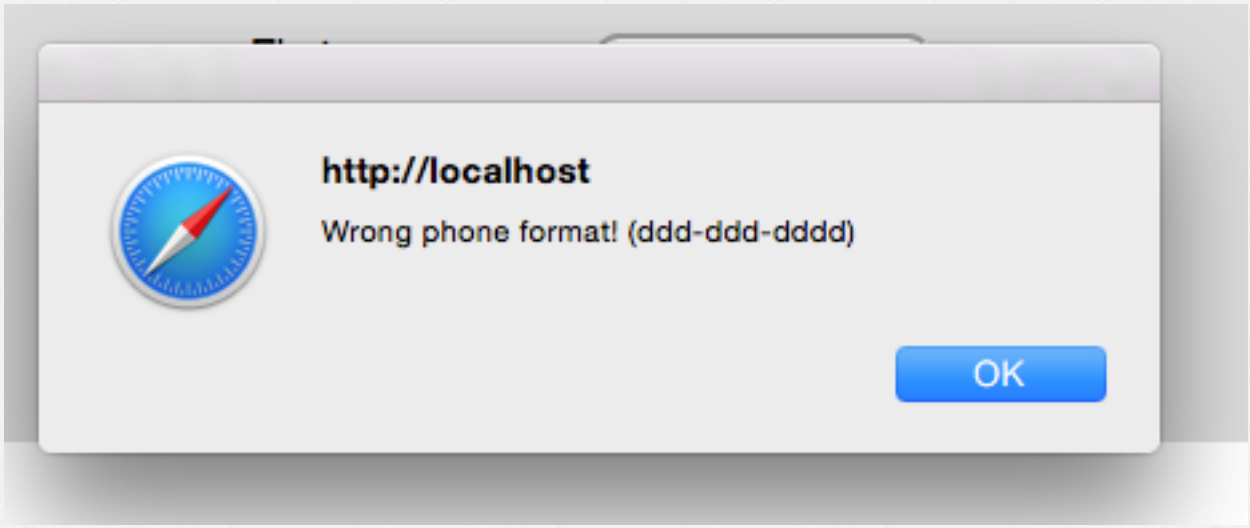
First name: milad

Password: ••••••

Re-enter Password: •••••• Passwords don't match !

Phone:

submit form



basic idea

- use the event `onSubmit` inside the form element:

```
<form
onsubmit = "return validateform();">
</form>
```

- design the function `validateform()` such that it returns ***true*** if everything is alright and ***false*** otherwise.
- When the function returns **`true`**, the form is submitted to the script specified in the action field.
- When the function returns **`false`**, the form is not submitted.
- Note that the user needs to know where he/she has made mistakes.

matching an email

```
/^\w+([\.-]?\w+)*@\w+([\.-]?\w+)*(\.\w{2,3})+$/
```

matching a password of at least 6 characters with capital, lowercase and number

```
/^(?=.{6})(?=.*?[a-z])(?=.*?[A-Z])(?=.*?[0-9])/
```

The ?= means that we care about the match but we don't care where. Order is not important

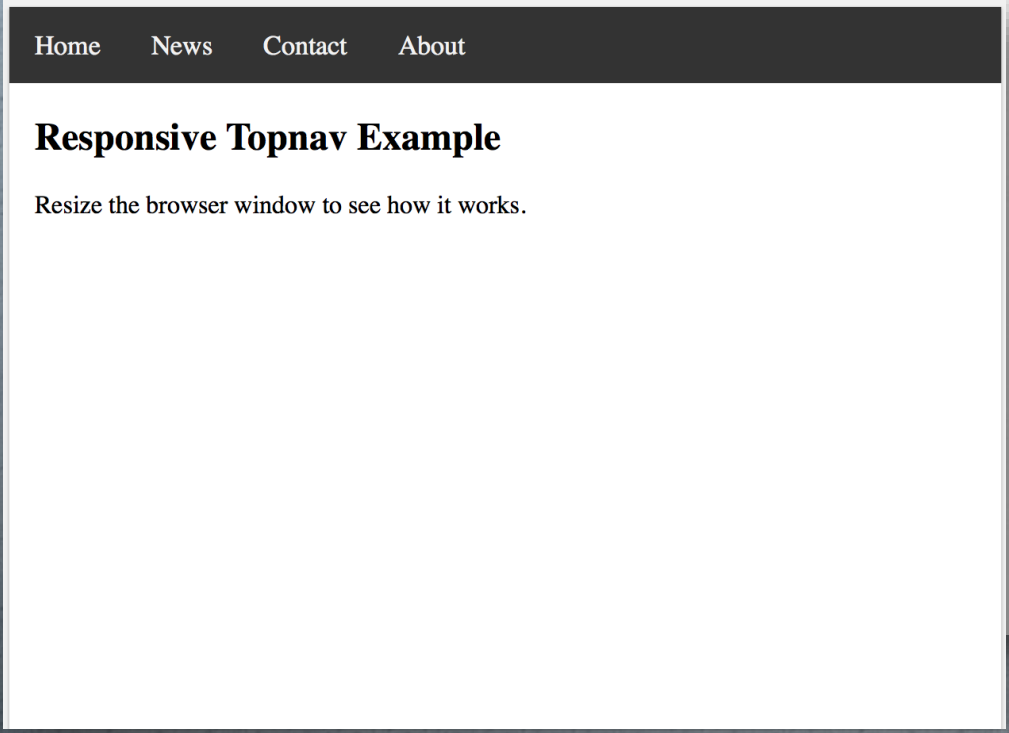
start of optional part

Example 1 (optional)

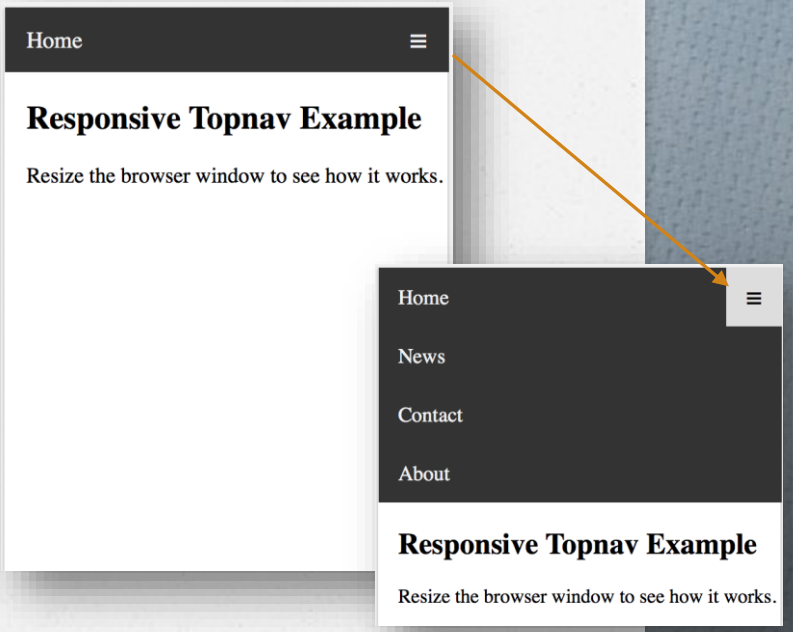
Responsive navigation menu

- Show navigation bar on big screens (laptops, desktops) and a hidden menu that can be expanded on small screens (phones, tablets)

Desktop version



Mobile version



Example 1 (optional)

step 1: create the nav bar

```
<div class="topnav" id="myTopnav">
  <a href="#home">Home</a>
  <a href="#news">News</a>
  <a href="#contact">Contact</a>
  <a href="#about">About</a>
  <a href="javascript:void(0);" class="icon"
onclick="myFunction()">&#9776;</a>
</div>
```

tell the browser not to follow link when it's clicked. we do this when we have a js function on the link

symbol code of the 3 dashes

- notice the class `topnav and icon`

Example 1 (optional)

step 2: CSS for the class topnav mentioned before

```
/* Add a black background color to the top navigation */
.topnav {
    background-color: #333;
    overflow: hidden;
}

/* Style the links inside the navigation bar */
.topnav a {
    float: left;
    display: block;
    color: #f2f2f2;
    text-align: center;
    padding: 14px 16px;
    text-decoration: none;
    font-size: 17px;
}
```

Example 1

(optional)

step 2: CSS for the class topnav (cont.)

```
/* Change the color of links on hover */
.topnav a:hover {
    background-color: #ddd;
    color: black;
}

/* Hide the link that should open and close the topnav on
small screens */
.topnav .icon {
    display: none;
}
```

Example 1

(optional)

step 3: Add the media queries to CSS

```
/* When the screen is less than 600 pixels wide, hide
all links, except for the first one ("Home"). Show the
link that contains should open and close the topnav
(.icon) */

@media screen and (max-width: 600px) {
  .topnav a:not(:first-child) {display: none;}
  .topnav a.icon {
    float: right;
    display: block;
  }
}
```

Example 1

(optional)

step 3: Add the media queries to CSS (cont.)

```
/* The "responsive" class is added to the topnav with
JavaScript when the user clicks on the icon. This class
makes the topnav look good on small screens (display the
links vertically instead of horizontally) */

@media screen and (max-width: 600px) {
  .topnav.responsive {position: relative;}
  .topnav.responsive a.icon {
    position: absolute;
    right: 0;
    top: 0;
  }
  .topnav.responsive a {
    float: none;
    display: block;
    text-align: left;
  }
}
```

notice the introduction of class called responsive that we haven't used before.
This will be added programmatically in JavaScript.

Example 1 (optional)

Step 4: Javascript

```
/* Toggle between adding and removing the "responsive"
class to topnav when the user clicks on the icon */
function myFunction() {
    var x = document.getElementById("myTopnav");
    if (x.className === "topnav") {
        x.className += " responsive";
    } else {
        x.className = "topnav";
    }
}
```