

CENG420

Chapter 4 JavaScript

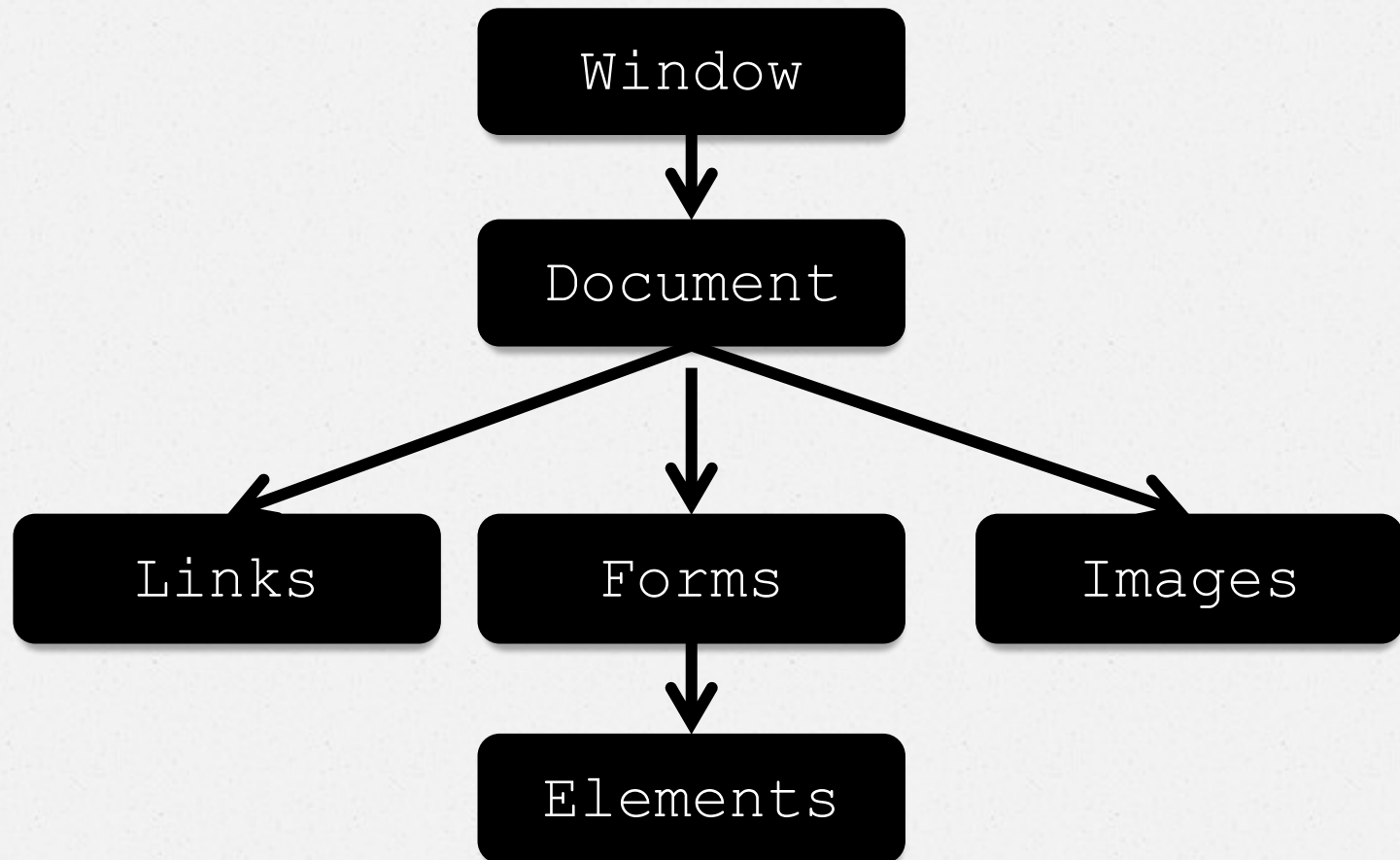
Part 2



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

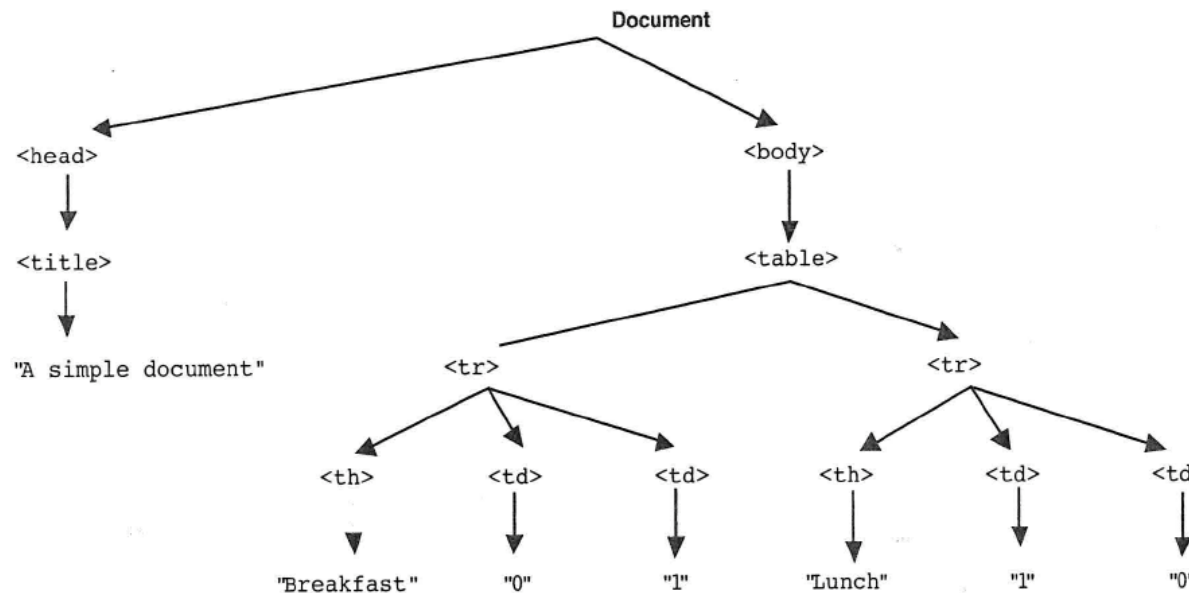
DOM

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>
```





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

- o Use the DOM forms array and elements array:

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

- o This will access the first element of the first form of the document → what is inside the variable dom now?
- o 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

- o Use "id" and getElementById()

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

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

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

Accessing radio/checkboxes

- o 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

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

Manipulating HTML Elements

To access an HTML element from JavaScript,

- o you can use the `document.getElementById(id)` method.
- o Use the *id* attribute to identify the HTML element
- o 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>
```

My First Web Page

Paragraph changed.

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>
```

My First JavaScript

JavaScript can change the content of an HTML element:

Click Me!

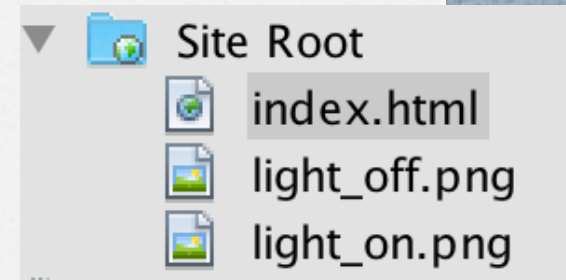
This is a demonstration.



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



Turn on



Turn off



Turn on

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>
```

My First JavaScript

JavaScript can change the style of an HTML element.

Click Me!

My First JavaScript

JavaScript can change the style of an HTML element.

Click Me!

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

o Assume you have 2 CSS classes myclass1 and myclass2.

o 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

o **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:

Creating elements

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

Example

- o Assume we want to add a <select> inside <div id="mydiv"> 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;
```




Events & Event handling

Events

- o An event can happen to any element in the document
- o Events like clicking, blurring, focus, loading, ...
- o 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
<code>mousedown</code>	<code>onmousedown</code>
<code>mousemove</code>	<code>onmousemove</code>
<code>mouseout</code>	<code>onmouseout</code>
<code>mouseover</code>	<code>onmouseover</code>
<code>mouseup</code>	<code>onmouseup</code>
<code>reset</code>	<code>onreset</code>
<code>select</code>	<code>onselect</code>
<code>submit</code>	<code>onsubmit</code>
<code>unload</code>	<code>onunload</code>

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;
```




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



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>

    </table>
</form>
```

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

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

Please input your details below

First name

Password

Password too short

Re-enter

Password

Phone

submit form

Please input your details below

First name

Password

Re-enter

Passwords don't match !

Password

Phone

submit form



http://localhost

Wrong phone format! (ddd-ddd-dddd)

OK

basic idea

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

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

- o design the function `validateform()` such that it returns **true** if everything is alright and **false** otherwise.
- o When the function returns **true**, the form is submitted to the script specified in the action field.
- o When the function returns **false**, the form is not submitted.
- o 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



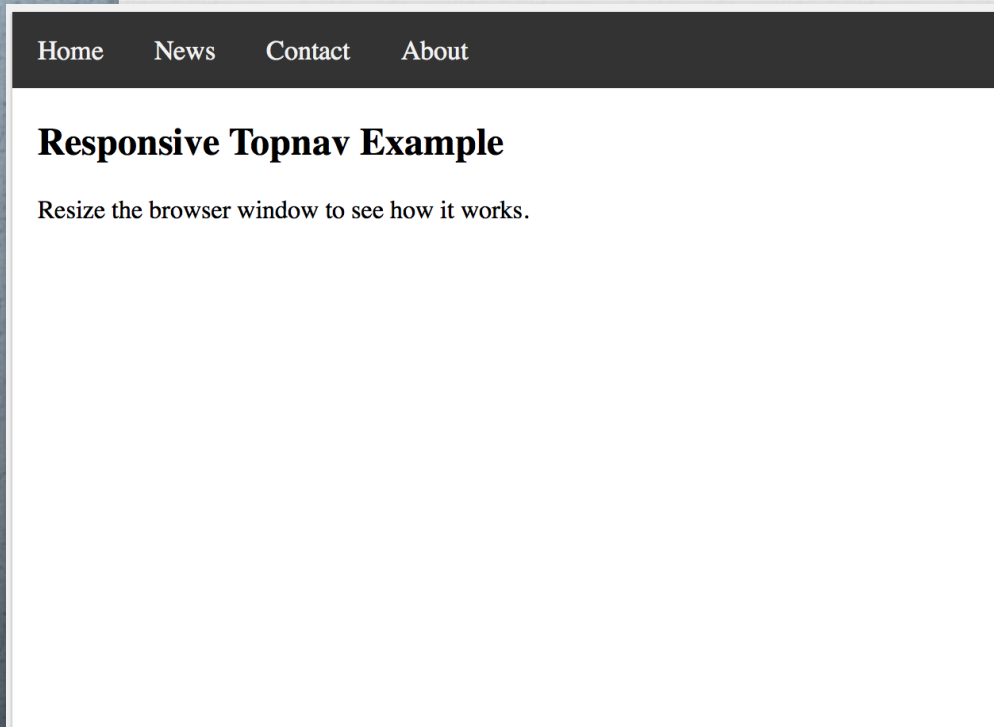
start of optional part

Example 1 (optional)

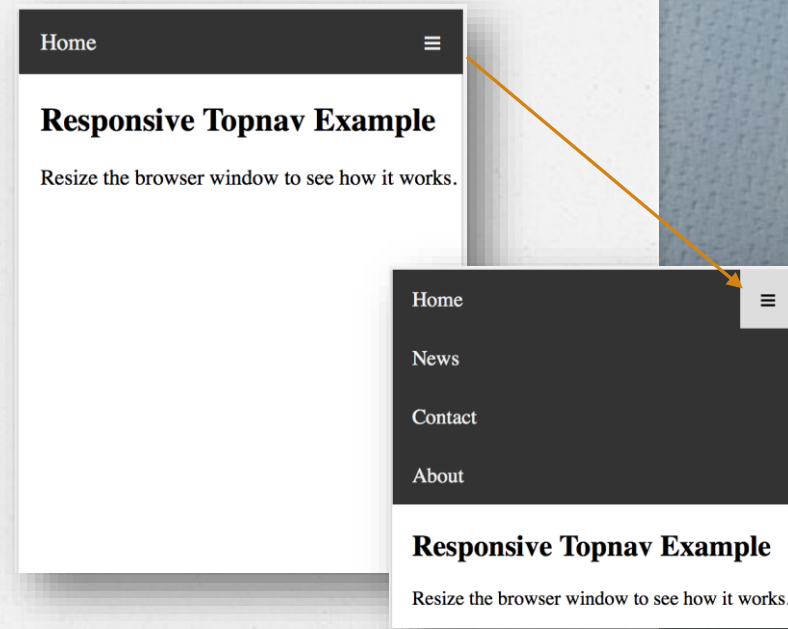
Responsive navigation menu

- 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";
    }
}
```