Session 4. JavaScript in the Browser

Programació Multiplataforma i Distribuïda

Grau d'informàtica. EPSEVG

Setembre 2019

Jordi Esteve jesteve@cs.upc.edu

JavaScript. Loading scripts in the Browser

JavaScript in Web pages allows to do actions triggered by buttons, receive events, change the content or the styles of the page, ...

Like CSS, JavaScript code can be loaded in two ways:

1. JavaScript code in an **external file**:

```
<script src="logic.js"></script>
```

2. JavaScript code **inside the body** of the page:

```
<body>
  <script>
    // JavaScript code
  </script>
</body>
```

- Slower (not stored in the browser cache)
- Not reusable by other Web pages

JavaScript. Example of script

Create the **index.html** and **logic.js** files, copy this content, save them and load index.html in your Web browser.

```
<!DOCTYPE html>
<html lang="en">
<head>
	<meta charset="UTF-8">
	<title>First Web page with JavaScrip</title>
	<script src="logic.js"></script>
</head>
<body>
	<p><strong>Date-Time: </strong><span id="time"></span></p>
	<button onclick="show_date()">Current date</button>
</body>
</html>
```

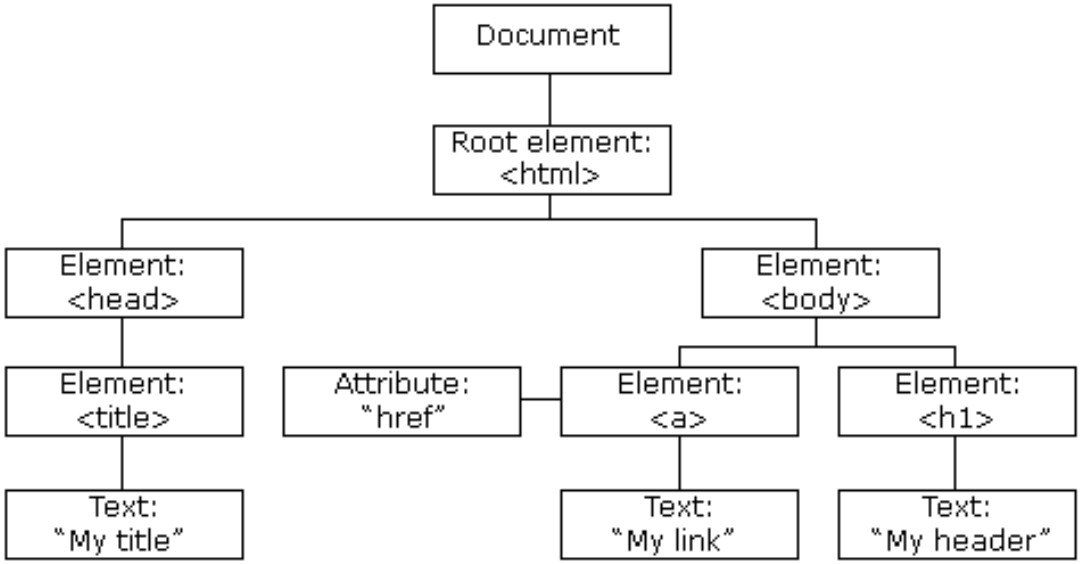
```
setInterval(
	() => {
		document.getElementById("time").innerHTML = new Date().toUTCString();
	}, 1000); // Calls this anonymous function every 1000 miliseconds

function show_date() {
	let d = new Date();
	alert(d.toDateString()); // Shows a pop-up window with the current date
}
```

JavaScript. Web browser objects

The Web browsers offers two objects to be used by JavaScript:

- **window**: Allows JavaScript control the Browser Object Model (**BOM**), e.g. window width and height, open/close/move/resize window, … There are not official standards.
- **document**: Allows JavaScript control the Document Object Model (**DOM**), the HTML document shown in the Web browser. DOM is a standard for access to the HTML **elements** and the **properties**, **methods** and **events** of these elements.

HTML DOM model is a tree of objects



JavaScript. DOM document methods and properties

Finding HTML Elements

document.getElementById(*idname*)	Returns the element whose id is *idname*
document.getElementsByClassName(c*name*)	Returns an array of elements having *cname* class
document.getElementsByName(*name*)	Returns an array of elements having *name* name (useful in forms: the input fields have different names)
document.getElementsByTagName(tag*name*)	Returns an array of elements having *tagname* tag
document.querySelectorAll(css_selector)	Returns array of elements by CSS selector (#id1, p.cl2)

Creating Elements

document.createElement(*element*)	Create an HTML element

Some **document properties:**

.*characterSet*, .*title*, .*URL*: Returns the character encoding, title or the full URL of the document.

.*cookie*, .*forms*, .*images*, .*links*: Returns all the cookies, forms, images or links of the document.

More information in DOM Document.

JavaScript. DOM element methods and properties

Adding and deleting Elements

element.removeChild(*element*)	Remove an HTML element
element.appendChild(*element*)	Add an HTML element
element.replaceChild(*new, old*)	Replace an HTML element

Changing HTML Elements

element.innerHTML = *new html content*	Change the inner HTML of an element
element.setAttribute*(attribute, value)*	Change the attribute value of an HTML element
element.attribute = *new value*	Change the attribute value of an HTML element

Some **element properties** we can get or set:

element.classList, element.id, element.style, element.title, element.value, ...

```
document.getElementById("id1").classList.add("c1"); // Adds "c1" class to the element with id "id1"
document.getElementsByClassName("c2")[0].style.visibility = "hidden"; // Hides 1st elem. of "c2" class
```

More information in DOM Element.

JavaScript. Example of script that does not work

Create the **index.html** and **logic.js** files, copy this content, save them and load index.html in your Web browser.

Why the page does not work? Why it does not show the fruit list?

```
<!DOCTYPE html>
<html lang="en">
<head>
	<meta charset="UTF-8">
	<title>Second Web page with JavaScrip</title>
	<script src="logic.js"></script>
</head>
<body>
	<ul id="l1"></ul>
	<p>The clicked item is <em><span id="c_item"></span></em></p>
</body>
</html>
```

```
function show_item(it) {
	document.getElementById("c_item").innerHTML = it.innerHTML;
}

let list = ["Apple", "Orange", "Pineapple", "Coconut"];
let l = "";
for(let e of list)
	l += `<li onclick="show_item(this)">${e}</li>\n`; // this = special variable containing itself
document.getElementById("l1").innerHTML = l;
```

JavaScript. Example of script fixed

The problem is that the JavaScript is executed before the Web page was loaded: The ul item with id="l1" was not created when the script tries to add list items.

To solve this problem we could move the initial code inside a function and call it when the page is loaded using the **onload** attribute in the body tag:

```
...
<body onload="init()">
...
```

```
function show_item(it) {
	document.getElementById("c_item").innerHTML = it.innerHTML;
}

function init() {
	let list = ["Apple", "Orange", "Pineapple", "Coconut"], l = "";
	for(let e of list)
		l += `<li onclick="show_item(this)">${e}</li>\n`;
	document.getElementById("l1").innerHTML = l;
}
```

JavaScript. Memory game exercise (I)

Improve the memory game created at end of session 1. You must create the initial board of the memory game with the cards allocated in a random places. If you reload the Web page of the game, different random boards must be shown.

Remember that **initial_cards(n)** returns an array of **n** elements containing **n/2** pairs of cards in random positions.

Tip:

- If you have the figures of the game stored in external files (figure1.svg, figure2.svg, figure3.svg, …), you could add <image> tags inside the html definition of the board with source files names created from the result of the **initial_cards(n)** function.
- If you have the figures of the game defined inside the html file, you could store them in an array (like the fruits in the previous exercise). In the html definition of the board you could pick the figures from the array depending the result of the **initial_cards(n)** function.

JavaScript. Memory game exercise (II)

1. When we start the memory game, all cards must be hidden. You could create this style in your CSS file

   ```
   .hidden {opacity: 0;}
   ```

   and add the class **hidden** to all the cards.

2. Add to all the cards an **onclick** attribute that calls a function. This function must remove the hidden class to the clicked card so the card will become visible. Use the special value **this** to pass the clicked element. See the fruits example or this example.

3. The card must remain only one second visible, then it must be hidden again. In the previous function that makes visible a card, you could call

   setTimeout(hide_card, 1000); // Calls hide_card function after 1000 miliseconds