Introduction to Web Programming

Chap. 5 / Javascript, part II

Anne Jeannin-Girardon, PhD | anne.jeannin@unistra.fr
Associate Professor, University of Strasbourg

Objects in JavaScript

- Objects = data (properties) + functions (methods)
- Handy way of representing things
- Properties: `key: value`

Functions

```
var hotel = {
  name: "Good Sleep Inn",
  rooms: 30,
  booked: 20,
  spa: true;
  roomType: ["double", "single"],

  checkAvailability: function(){
    return
      this.rooms - this.booked;
  }
};
```

properties

method

Dot notation

```
var hotelName = hotel.name;
var freeRooms = hotel.checkAvailability();
```

Member operator

```
var hotelName2 = hotel['name']; // properties only

hotel.name = "Sleepwell Inn";
```

Using a constructor

- Constructor = function used to create an object
- Each object created is an instance of the object

POO coding convention : class names start with an uppercase letter

Constructor

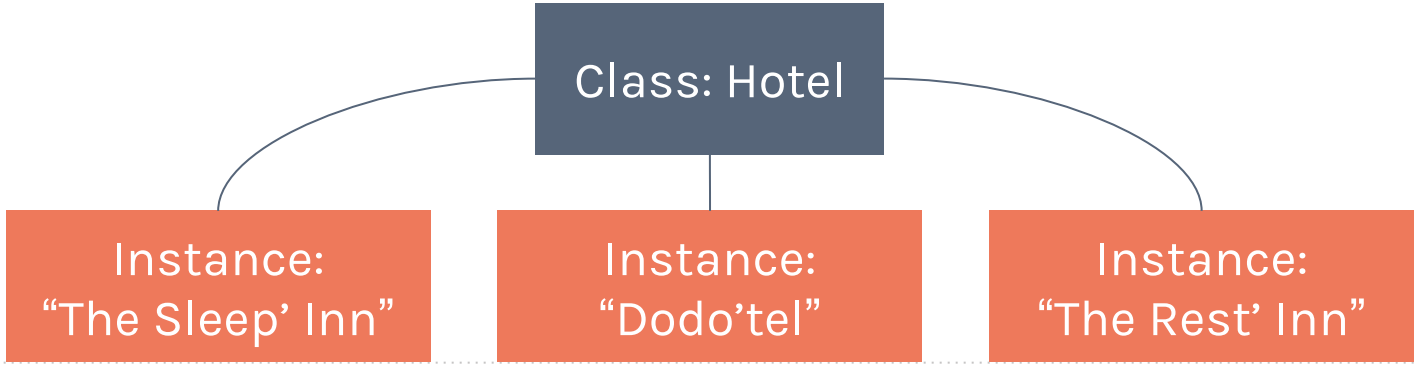
```
function Hotel(name, rooms, booked){
  this.name = name;
  this.rooms  = rooms;
  this.booked = booked;

  this.checkAvailability = function(){
    return
      this.rooms - this.booked;
  }
};

var hotel = new Hotel("Sleep Inn", 30, 20);
```

The keyword `this`

- Recall that a class is a blueprint
- You can instantiate (build) as many objects of a given class as you want
- Each instance has its own copies of properties and methods
- If you want to access one instance in particular, use `this` (inside the class definition) : `this` is a reference to a particular instance



The operator instanceof

instanceof

```
var h = new Hotel("Sleep Inn", 30, 20);

if (h instanceof Hotel){
  console.log("It’s an hotel!");
}
```

Using the object document

- The HTML document is an object itself
- It as properties and methods
- We'll use it to access the elements of our web pages -- because we have a nicely structured HTML document ☺

Guess what those methods return:

- `document.getElementById("content");`
- `document.getElementsByTagName("p");`
- `document.getElementsByClassName("myClass");`

- The element identified by the id "content"
- All paragraph elements (stored in an array !)
- All elements having the class "myClass" (stored in an array!)

Manipulating HTML elements

Accessing HTML element

```
/* Assuming we have the following in the
HTML document :
<div id="test"></div>
*/
var elt = document.getElementById("test");
elt.innerHTML = "Hello world!";

var attrName  = elt.attributes[0].name;
var attrValue = elt.attributes[0].value;
console.log(attrName + " " + attrValue);
console.log(elt.innerHTML);
```

Monitoring events

- You can trigger behaviors if a given event occurs in the page (clicking, hovering, …)
- More specifically, you trigger behavior by monitoring elements in your page
- We talk about event listeners

Mouse events: https://developer.mozilla.org/en-US/docs/Web/Events#Mouse_events

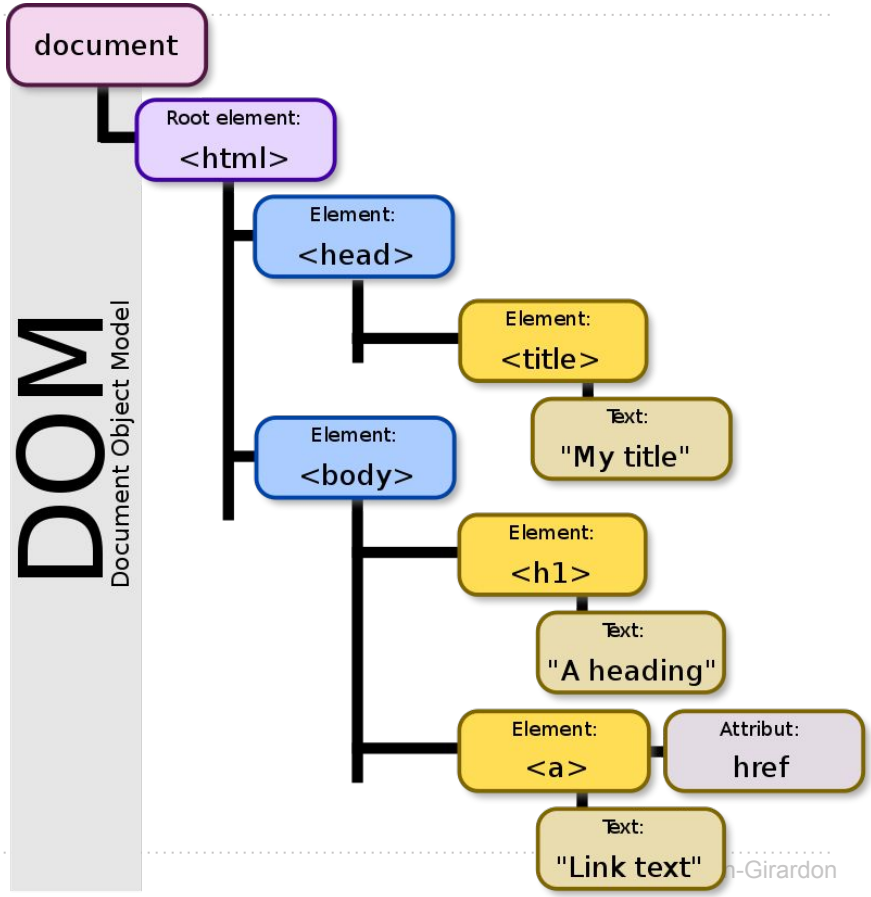
Event listener

```
var x = document.getElementById("myButton");
x.addEventListener("mouseover", function1);
x.addEventListener("click",function2);
x.addEventListener("mouseout", function3);

function function1() {
    // behavior when the button
    // has been hovered
}
function function2() {
    // behavior when the button
    // has been clicked
}
function function3() {
    // behavior then the mouse
    // leaves the button
}
```

Document Object Model (DOM)

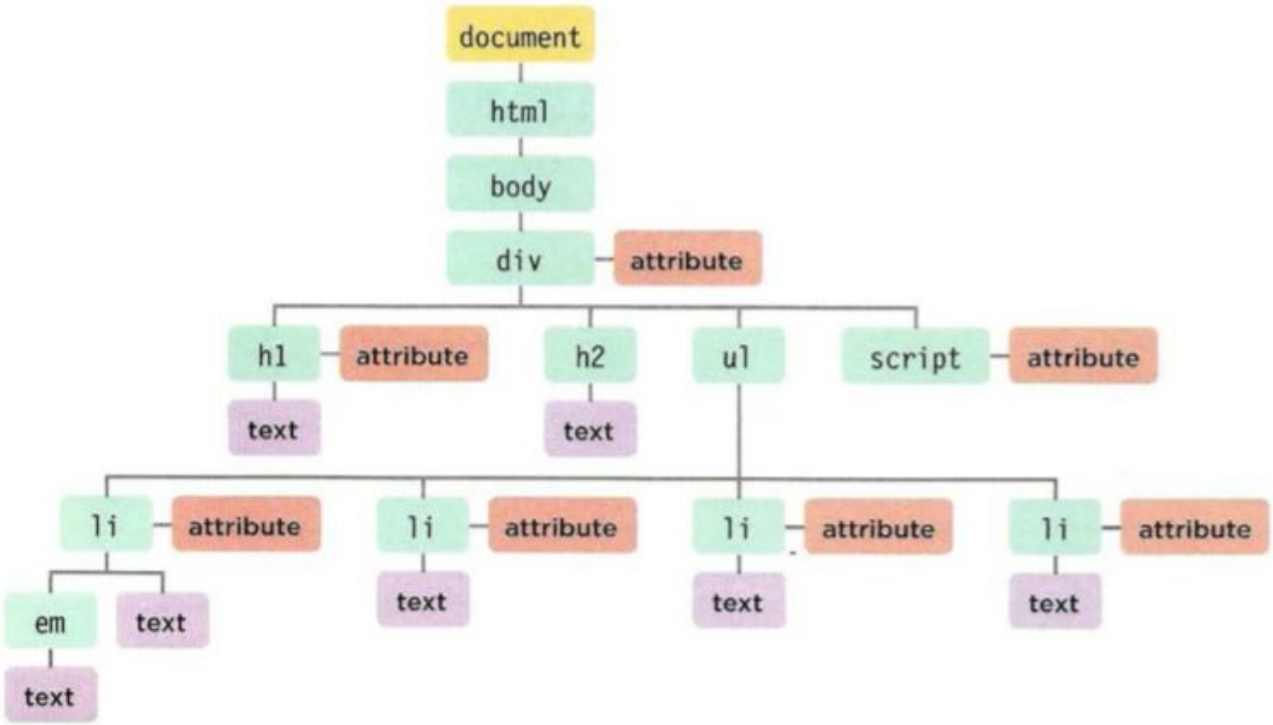
- W3C standard
- Starts from a root (“document”) and represent the document as a hierarchy (parents, children, siblings)
- All HTML elements are seen as objects (so they have attributes and methods)
- Understanding this hierarchy will allow you to use JS to access the elements of your document



Nodes in the DOM

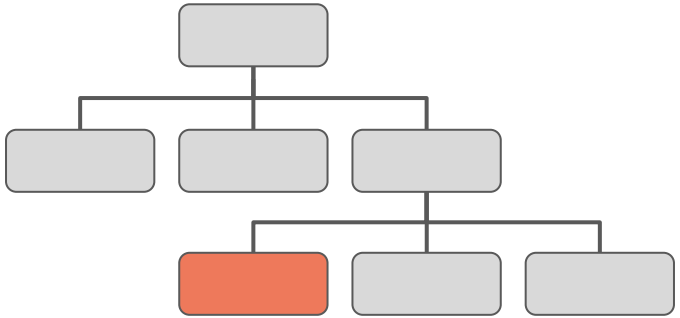
- The document node: this is the root of your document. We have seen this object during the last lecture, remember what we did ?
- Element nodes: the HTML elements that describe the structure of your document (can you give me some examples of such elements ?)
- Attribute nodes: the attributes that are carried by elements (can you give me an example of attribute for an image element ?)
- Text nodes: the text contained in an element. Those are the leaves of the tree (they cannot have children nor siblings)

Example of DOM



Accessing individual element nodes

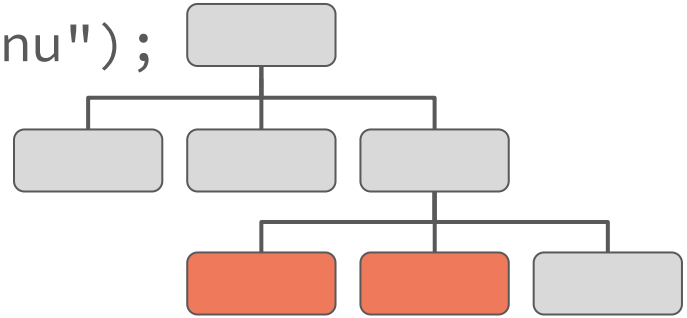
- `document.getElementById("myID");`
  Uses the id attribute
- `document.querySelector("li.menu");`
  Returns the first elements matching the query



Accessing multiple elements

These methods returns arrays of elements

- `document.getElementsByClassName("title");`
  Select *all* elements carrying the specified class
- `document.getElementsByTagName("p");`
  Select *all* elements having the specified tag
- `document.querySelectorAll("li.menu");`
  Select *all* elements matching the query



Example : modifying the class of all paragraphs

Accessing multiple elements

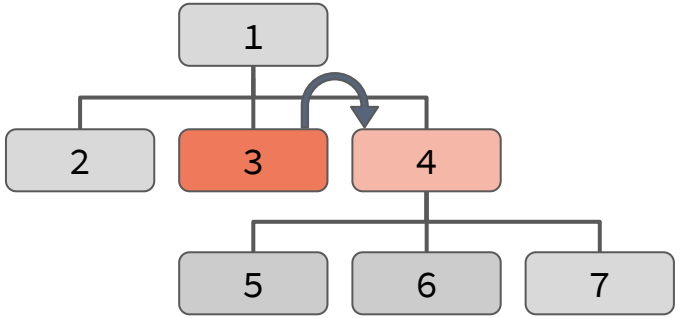
```
var pars = document.getElementsByTagName("p");

for (var i = 0; i < pars.length; i++){
  pars[i].setAttribute("class", "myClass");
}
```

Traversing between nodes

```
var elt = document.getElementById("myID");
```

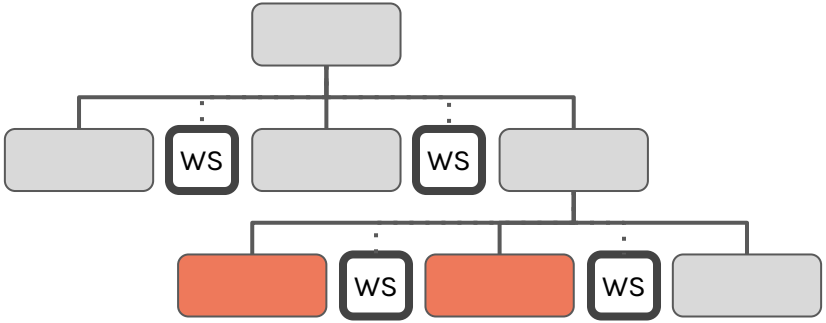
- `var parent = elt.parentNode;`
- `var prev   = elt.previousSibling;`
- `var next   = elt.nextSibling;`
- `var fChild = elt.firstChild;`
- `var lChild = elt.lastChild;`



No prev/next/children ? then it returns `null`

Beware of whitespaces !

- Most browsers consider whitespaces (spaces, carriage returns) as text nodes
- You must remember this when using the previous attributes so you know if what you get using them is what you expect or not



Modifying the HTML code

- Changing elements' attributes:

```
document.getElementById("myID").style.color = "blue";
document.getElementById("myPic").src = "newPic.png";
```

- Changing the content of elements (text+html)

```
document.getElementById("myElt").innerHTML = "<em>Hello!</em>";
```

- Changing the content of elements (text only)

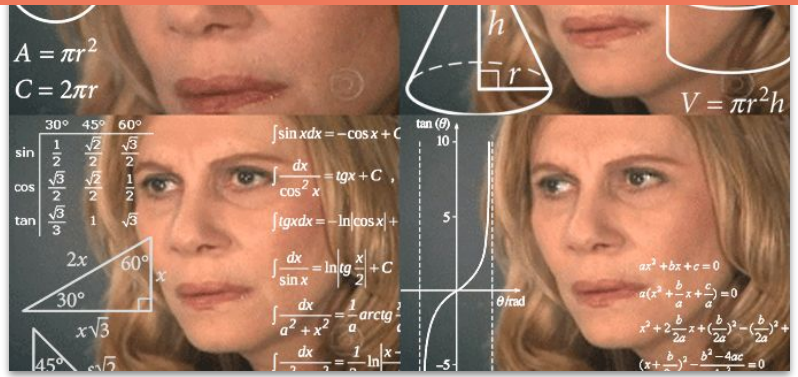
```
document.getElementById("myElt").textContent = "Hello!";
```

Cascading accesses

What does this instruction return ? (draw the DOM tree to illustrate your answer)

```
document.getElementsByTagName("p")[0].firstChild.nextSibling.textContent;
```

The textual content of the next sibling of the first child of the paragraph 0 of the document



Exercises

- Consider the following HTML snippet: `<p>Hello world!</p>`
  1. What is the html node ?
  2. What is the value of the text node of this element ?
- Consider the following HTML snippet: `<p>Hello <em>world</em>!</p>`
  What is the value of the text node of this element ?
- What does the following instruction return ?

```
document.getElementById("content").lastChild.prevSibling.parentNode.id;
```

Exercise

Draw the DOM tree corresponding to the following HTML structure

HTML structure

```
<!DOCTYPE HTML>
<html>
<head>
  <title>Draw me a DOM tree !</title>
</head>
<body>
  <header>A nice exercise</header>
  <main>
    <p>Lorem ipsum</p>
  </main>
  <footer>ProgWeb1 2019</footer>
</body>
</html>
```

Modifying the DOM: adding/removing elements

Steps:

(1) Create the element

(2) Create a text node

(3) Attach the text node to the element

(4) Attach the element to the document

Modifying the DOM

```
/* Create a paragraph element and append it (as the
 * last child) to the element identified by “myParent”
 */

var elt = document.createElement("p");
var content = document.createTextNode("Hello world!");
elt.appendChild(content);
document.getElementById("myParent").appendChild(elt);
```

Where can I attach new elements ?

- To a parent, as the last child: `node.appendChild(newNode)`
- Before a given element: `node.insertBefore(newNode, existingNode)`
- Substitute element: `node.replace(newNode, oldNode)`
- Remove element (use with caution): `node.removeChild(childNode)`

Modifying the DOM

```
<!DOCTYPE html>
<html>
  <body>
  <ul id="myList1"><li>JS1</li><li>JS2</li></ul>
  <ul id="myList2"><li>HTML1</li><li>HTML2</li></ul>
  <button onclick="myFunction()">Go !</button>

<script>
function myFunction() {
  var node =
        document.getElementById("myList2").lastChild;
  var list = document.getElementById("myList1");
  list.insertBefore(node, list.childNodes[0]);
}
</script>
  </body>
</html>
```

Application example: todo list

Sounds easy enough, right ?

Well…

Not all browsers support methods such as getElementsByTagName(), etc.

So basically, I just taught you how to produce non-portable JS code, right ?
Yeah, a bit… But it's important to learn Vanilla Javascript

Now, we can go further and learn how to use a library that will unify what we need (especially regarding DOM access) across all browser

Disclaimer

There's quite a lot of debate over the internets about jQuery v.s. Vanilla Javascript

I don't have a side. I don't want you to pick a side because of this Web Development class (wanna pick a side ? search the web about this)

I merely think that jQuery is a nice and easy way to learn how to use Javascript libraries.

If you want to find out more about this debate, here is a somehow neutral starting point : https://www.codementor.io/brainyfarm/jquery-vs-vanilla-javascript-deciding-on-what-to-use-6b79xdmrv

Chapter recap

- Notion of objects
- Seeing the HTML document as the object it is
- Formalizing the DOM
- DOM nodes
- Accessing DOM nodes
- Modifying the HTML code of a page
- Modifying the DOM