The DOM

Document-Object Model

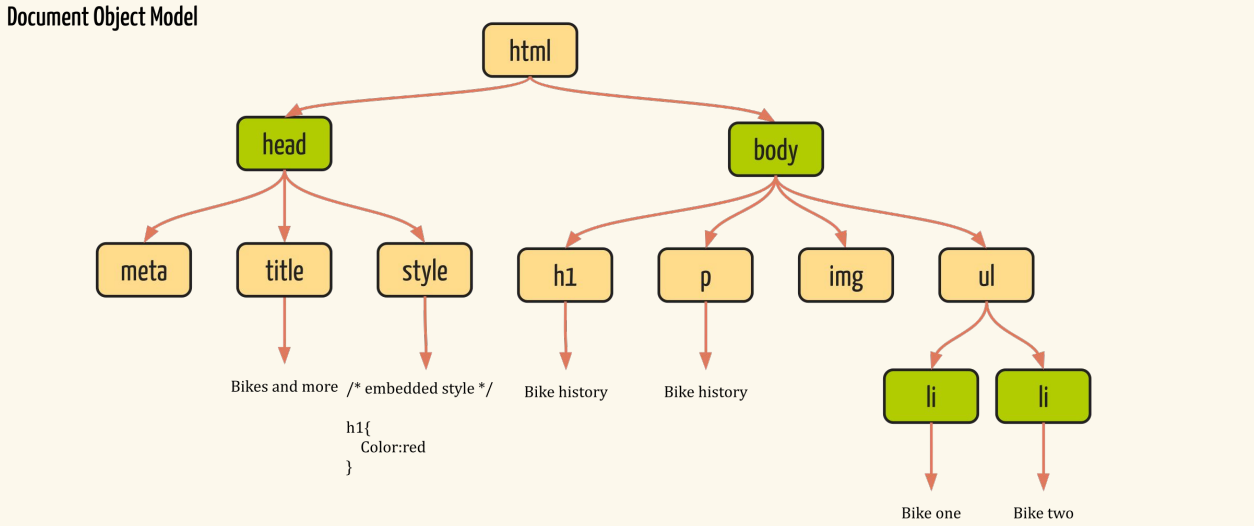
Shadi Lahham - Programmazione web - Frontend - Javascript

Accessing the DOM

What is the DOM?

- Document Object Model - a map of our HTML document
- The logical structure of an HTML document, how it is accessed and manipulated
- Elements in an HTML document can be accessed, changed, deleted, or added using the DOM

What is the DOM?



Inspecting the DOM

- Google Chrome or Mozilla Firefox
  - Right-click on a web page and select "Inspect Element"
  - Shortcuts: Cmd-Opt-I (Mac), Ctrl-Shift-I (Windows)
  - Shortcuts: F12

- Safari
  - Unlock the Develop Menu by opening Safari > Preferences > Advanced, and checking the box, "Show Develop menu in menu bar."
  - Access by same methods as Chrome (above)

Try it now on any web page

Accessing the DOM

The document object is globally available in your browser. It allows you to access and manipulate the DOM of the current web page:

1. **Find** the DOM node you want to change using an access method
2. **Store** this DOM node as a variable
3. **Manipulate** the DOM node
    - Change its attributes
    - Modify its styles
    - Give it new inner HTML
    - Append new nodes to it

DOM Access Methods

```
// Finding DOM nodes by id:
document.getElementById(id);

//Finding DOM nodes by tag name:
document.getElementsByTagName(tagName);

//Finding DOM nodes by class name:
document.getElementsByClassName(className);

//Finding DOM nodes by query selector:
document.querySelector(cssQuery);
document.querySelectorAll(cssQuery);
```

Selecting Nodes From the DOM

Javascript

```
// By Id
let hobbiesList =
document.getElementById('hobby-list');

// By Tag Name
let hobbies =
document.getElementsByTagName('li');

// By Class Name
let alsoHobbies =
document.getElementsByClassName('hobby');

// By CSS Query
let firstHobby = document.querySelector('ul
li.hobby');
let allItems = document.querySelectorAll('ul
li.hobby');
```

HTML

```
<ul id="hobby-list">
  <li class="hobby">Playing the banjo</li>
  <li class="hobby">Paddleboarding</li>
</ul>
```

Return values

getElementById() and querySelector()return a single element:

```
let hobbiesList = document.getElementById('hobby-list');
let firstHobby = document.querySelector('ul li.hobby');
```

getElementsByClassName(), getElementsByTagName(), and querySelectorAll() return a collection of elements.
It behaves like an array.

```
let catNames = document.querySelectorAll('ul li.catname');
let firstCatName = catNames[0];
```

Manipulating the DOM

Manipulating a node's attributes

You can access and change the attributes of a DOM node using dot notation.

```
<img id="lizzie" src="http://placekitten.com/200/300">

// Changing the src of an image:
let catImage = document.getElementById('lizzie');
let oldImageSource = catImage.src;
catImage.src = 'http://placekitten.com/300/200';

//Changing the className of a DOM node:
let catImage = document.getElementById('lizzie');
catImage.className = 'portrait';
```

Manipulating a node’s style

You can access and change the styles of a DOM nodes via the style property.
CSS property names with a "-" must be camelCased and number properties must have a unit.

```css
/* css */
body {
  color: red;
  background-color: pink;
  padding-top: 10px;
}
```

```javascript
// javascript - style is applied inline - not recommended to use
let pageNode = document.body;
pageNode.style.color = 'red';
pageNode.style.backgroundColor = 'pink';
pageNode.style.paddingTop = '10px';
```

Manipulating a node's inner HTML

Each DOM node has an innerHTML attribute. It contains the HTML of all its children:

```
let pageNode = document.body;
console.log(pageNode.innerHTML);
```

Set innerHTML to replace the contents of the node:

```
pageNode.innerHTML = "<h1>Oh, no! Everything is gone!</h1>"
```

Or add to innerHTML instead:

```
pageNode.innerHTML += "P.S. Please do write back.";
```

innerHTML vs. textContent

- If you're only changing the actual text of a node, textContent may be a better choice.
  - innerHTML
    - Works in older browsers
    - More powerful: can change code
    - Less secure: allows cross-site scripting (XSS)
  - textContent
    - Doesn't work in very old browsers
    - Faster: the browser doesn't have to parse HTML
    - More secure: won't execute code

Creating nodes

Creating DOM Nodes

The document object also allows us to create new nodes from scratch:

```
document.createElement('tagName');
document.createTextNode('text');
document.appendChild(childToAppend);
```

Exampe:

```
let body = document.body;
let newImg = document.createElement('img');
newImg.src = 'http://placekitten.com/400/300';
newImg.style.border = '1px solid black';
body.appendChild(newImg);

let newParagraph = document.createElement('p');
let newText = document.createTextNode('Squee!');
newParagraph.appendChild(newText);
body.appendChild(newParagraph);
```

Your turn

1.About Me

Start with this HTML

```
<!DOCTYPE html>
  <html>
    <head>
      <meta charset="utf-8"/>
      <title>About Me</title>
    </head>
    <body>
      <h1>About Me</h1>
      <ul>
        <li>Nickname: <span id="nickname"></span>
        <li>Favorites:  <span id="favorites"></span>
        <li>Hometown: <span id="hometown"></span>
      </ul>
    </body>
  </html>
```

Continues on next page >>>

1.About Me

- Add an external javascript file called main.js
- In JavaScript:
  - Change the body style so it has a font-family of "Arial, sans-serif"
  - Replace each of the spans (nickname, favorites, hometown) with your own information
  - Iterate through each li and change the class to "list-item"
  - Create a new img element and set its src attribute to a picture of you
  - Append that element to the page
- Add an external css file using Javascript
  - The external css file should make items with the .list-item class white, bold and with an orange background
  - The external css file should be applied after 4 seconds

2.The Book List

Use an array of books like this
You should have at least 4 books

```
let books = [
    {
      title: 'The Design of EveryDay Things',
      author: 'Don Norman',
      alreadyRead: false
    }, {
      title: 'The Most Human Human',
      author: 'Brian Christian',
      alreadyRead: true
    }
  ];
```

Continues on next page >>>

2.The Book List

- Create a *complete* webpage with a title, description and all other HTML tags
- In the body add an h1 title of "My Book List"
- In javascript, iterate through the array of books.
    - For each book, create HTML element with the book title and author and append it to the page
    - Use a ul and li to display the books
    - Add a url property to each book object that contains the cover image of the book
    - Add the image to the HTML using Javascript
    - Using javascript change the style of the book depending on whether you have read it or not
- Add an external css file that applies after 5 seconds
    - Now change the style of the book depending on whether you have read it or not using both css and javascript (the CSS should use a different color for read books)

3.DOM Detective

- Go to www.gog.com
- Use the devtools to view the DOM and write Javascript in the console
- Use the DOM access methods to find the following:
  - Every image on the page
  - The main menu at the top of the page
  - All the news items under "News"
  - The footer
  - All the social media links at the bottom of the page
- Produce a readme.md file with
  - snippets of your Javascript code
  - explanations of what which elements they select

4.Custom Detective

- Choose a news website that you like
- Use the devtools to view the DOM and write Javascript in the console
- Use the DOM access methods to find:
  - At least 10 different elements or collections of elements in the page
  - Choose interesting elements that require complex selectors to reach
- Produce a readme.md file with
  - A link to the website that you chose
  - snippets of your Javascript code
  - explanations of what which elements they select

Bonus

5.Arrivals

- Implement the arrivals page of an airport such as this one
  - Create a complete proper webpage with a title, description and all other HTML tags
  - Add Javascript and CSS files
  - Include as much detail as you can to each flight row
  - Add a Status to each flight. Status can be DEPARTING, DELAYED, ON_TIME, ARRIVED, etc
- Simulate a real arrivals list
  - The list should start empty and update every 10 seconds
  - Flights that have arrived should be removed after 60 seconds
  - Flights should change status in time. E.g. departing>on_time>delayed>arrived
  - Flights that are delayed should be displayed in red
  - New flights should be added to the bottom of the list
  - The list should be sorted by date and hour

6.Identity Hijack

Change the Stanford website using elements from the Berkeley website

- Brand and name
  - Find any elements with the word 'Stanford' and replace it with 'Berkeley'
  - Remember to change the title of the page as well
  - Replace any logos or symbols of Stanford University with Berkeley
- Colors
  - Find all elements with the 'Stanford' color(s) and replace them with the 'Berkeley' color(s)
- Links
  - Manually find all the links in the navigation area and replace them with references to the Berkeley website if there are similar pages there. Otherwise links should point to the Berkeley homepage
- Submit a Javascript file with all the changes

References

JavaScript HTML DOM

JavaScript DOM HTML

JavaScript DOM CSS

JavaScript DOM Elements

More detailed

The Document-Object Model