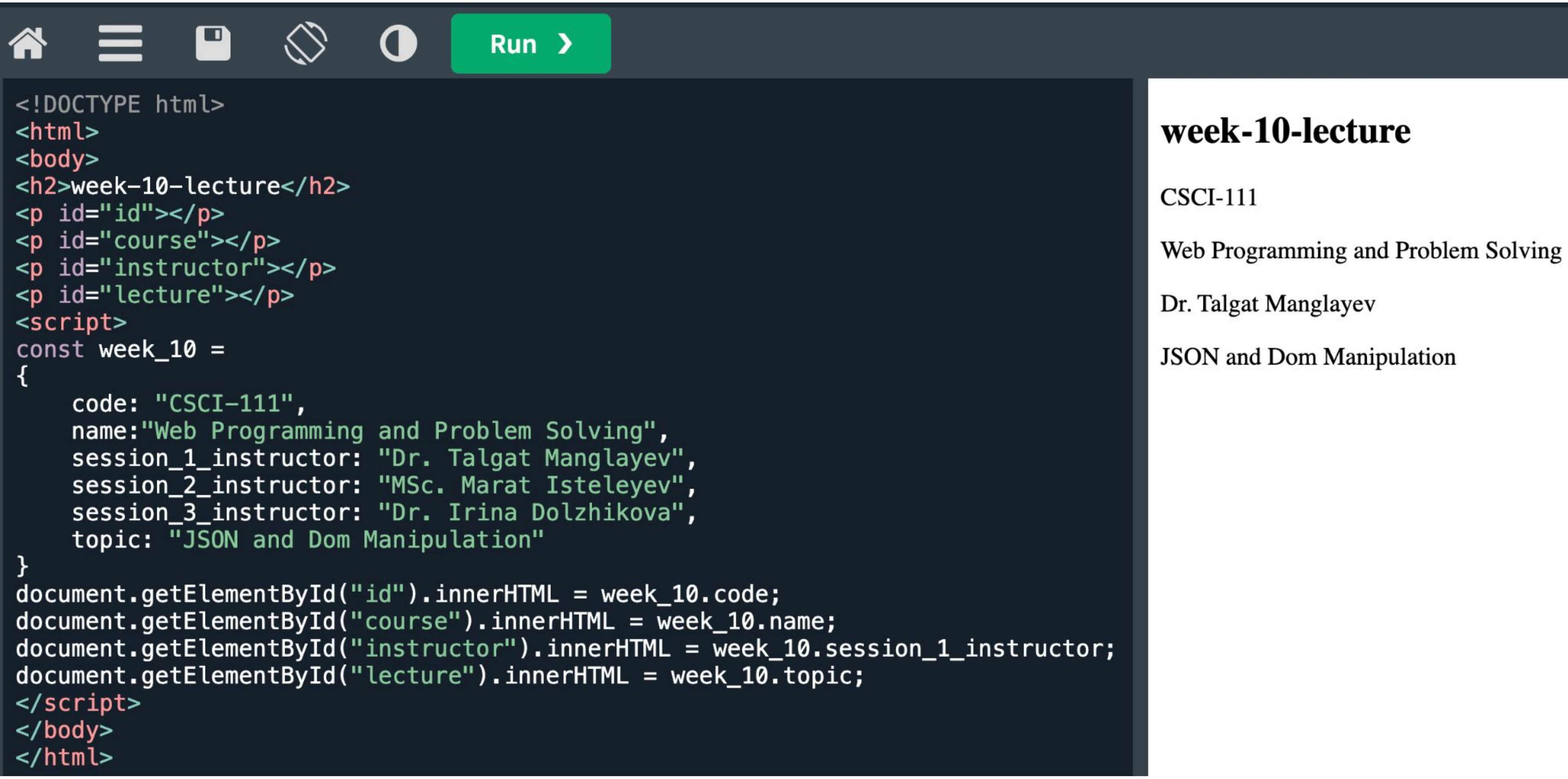


```
<!DOCTYPE html>
<html>
<body>
<h2>week-10-lecture</h2>
<p id="id"></p>
<p id="course"></p>
<p id="instructor"></p>
<p id="lecture"></p>
<script>
const week_10 =
{
    code: "CSCI-111",
    name:"Web Programming and Problem Solving",
    session_1_instructor: "Dr. Talgat Manglayev",
    session_2_instructor: "MSc. Marat Isteleyev",
    session_3_instructor: "Dr. Irina Dolzhikova",
    topic: "JSON and Dom Manipulation"
}
document.getElementById("id").innerHTML = week_10.code;
document.getElementById("course").innerHTML = week_10.name;
document.getElementById("instructor").innerHTML = week_10.session_1_instructor;
document.getElementById("lecture").innerHTML = week_10.topic;
</script>
</body>
</html>
```

week-10-lecture

CSCI-111

Web Programming and Problem Solving

Dr. Talgat Manglayev

JSON and Dom Manipulation

Content

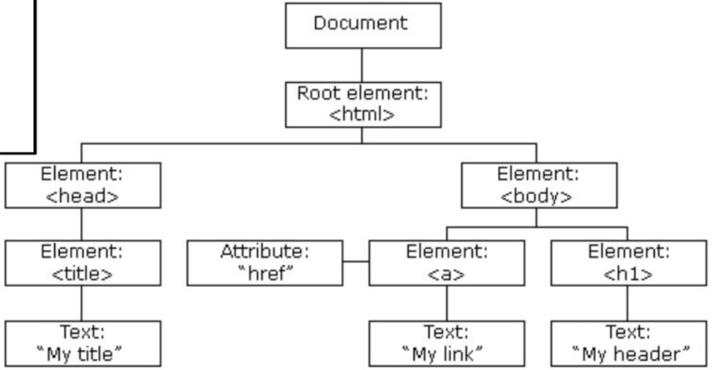
- What is Dom?
- Finding elements
- Element manipulation
- Content manipulation
- Attribute manipulation
- Style manipulation
- Class manipulation

What is Dom?

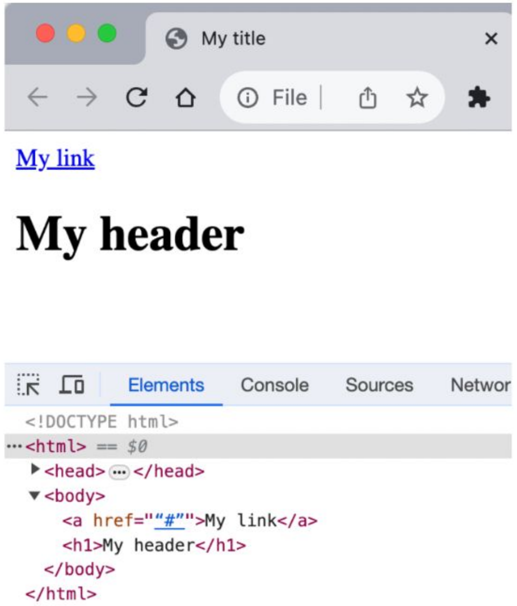
HTML view

```
<!DOCTYPE html>
<html>
  <head>
    <title>My title</title>
  </head>
  <body>
    <a href="#">My link</a>
    <h1>My header</h1>
  </body>
</html>
```

Graphical view



Browser view



My link

My header

```
<!DOCTYPE html>
…<html> == $0
  ▶<head>…</head>
  ▼<body>
      <a href=""#"">My link</a>
      <h1>My header</h1>
    </body>
</html>
```

What is HTML Dom?

Standard object model and programming interface for HTML document.

DOM defines:

- The HTML elements as objects
- The properties of all HTML elements
- The methods to access all HTML elements
- The events for all HTML elements

Document Object

The main object in the DOM.

All the elements are accessible via document object.

Main methods to access elements:

- Finding elements
- Creating elements
- Adding elements
- Deleting elements

Finding elements

The elements in DOM can be found by element's

id: `document.getElementById(<element_id>)`

tag: `document.getElementsByTagName(<tag_name>)`

class name:`document.getElementsByClassName(<class_name>)`

CSS selector: `document.querySelectorAll(<CSS selector>)`

Note that the last three methods return an array of objects.

Element manipulation

The main methods to work with elements are:

```
document.createElement(<element>)

document.removeChild(<element>)

document.appendChild(<element>)

document.replaceChild(<new_element, old_element>)
```

Element manipulation

Example: Adding a header element to the body

```
// print the initial document object
console.log(document)
// create an HTML element - h1
let h1 = document.createElement("h1")
// append the h1-element to the bodyelement
document.body.appendChild(h1)
// print the final document object
console.log(document)
```

Note that the header element has no text, i.e. it is empty

Content manipulation

Two basic properties to manipulate content of the elements: innerText, innerHTML.

```
// print the initial document object
console.log(document)
// create an HTML element - h1
let h1 = document.createElement("h1")
// create an HTML element - h1
h1.innerText = "<i>Header Text</i>"
// append the h1-element to the body-element
document.body.appendChild(h1)
// print the final document object
console.log(document)
```

NOTE:
innerHTML - the content is treated as content, i.e. properly decoding HTML tags.
innerText - the content is treated as text.

Attribute manipulation

The attribute of an HTML element can be accessed and manipulated using:

```
getAttribute(<attribute_name>)
setAttribute(<attribute_name>, <attribute_value>)

// create an HTML element - a
let a1 = document.createElement("a")
a1.setAttribute("href", "https://w3schools.com")
a1.innerText = "W3Schools"
document.body.appendChild(a1)
```

Note: getAttribute() methods returns null if there is no requested attribute

Style manipulation

To change CSS styles of an element, style property (attribute) can be used.

CSS property with dash are converted to camel case:

```
background-color → backgroundColor
```

```
let p1 = document.createElement("p")
p1.innerText = "This is the first paragraph!"
p1.style.color = "red"
p1.style.backgroundColor = "yellow"
document.body.appendChild(p1)
```

Class manipulation

The element might define several classes which can be accessed using `classList` property

`classList` returns the list of all classes

`classList` itself has methods: `add()`, `remove()` and `toggle()` a class

```
// add and remove a class to the element
let btn1 = document.getElementById("btn1")
console.log(btn1.classList)
btn1.classList.add("btn")
console.log(btn1.classList)
btn1.classList.remove("btn")
console.log(btn1.classList)
```

Objects

JSON

Summary

- DOM is a standard way to work with HTML document
- Document object is used to access other elements
- JavaScript allows to access and manipulate elements, their content, attributes and styles