CSS Selectors

1. Universal Selector

Selects all elements on the page.

```
* {
  margin: 0;
  padding: 0;
}
```

2. Type Selector (Element Selector)

Selects elements by their tag name.

```
p {
  font-size: 16px;
}
```

3. Class Selector

Selects elements with a specific class attribute.

```
.card {
  box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
}
```

4. ID Selector

Selects an element with a specific ID (should be unique).

```
#header {
  background-color: #333;
}
```

5. Group Selector

Applies the same styles to multiple elements.

```
h1, h2, h3 {
  font-family: 'Segoe UI', sans-serif;
}
```

6. Descendant Selector

Selects elements that are nested inside a specified ancestor.

```
nav ul li {
  list-style: none;
}
```

7. Child Selector (>)

Selects direct child elements only.

```
div > p {
  color: blue;
}
```

8. Adjacent Sibling Selector (+)

Selects the element that is immediately next to the specified element.

```
h2 + p {
  margin-top: 0;
}
```

9. General Sibling Selector (~)

Selects all siblings after a specified element.

```
h2 ~ p {
  color: gray;
}
```

10. Attribute Selector

Selects elements based on the presence or value of attributes.

```
input[type="text"] {
  border: 1px solid #ccc;
}
```

Other examples:

```
a[target]           /* selects links with a target attribute */
img[alt~="icon"]    /* selects images whose alt contains "icon" */
```

11. Pseudo-class Selector

Selects elements based on their state or position.

```
a:hover {
  color: red;
}

li:first-child {
  font-weight: bold;
}

input:focus {
  outline: none;
}
```

12. Pseudo-element Selector

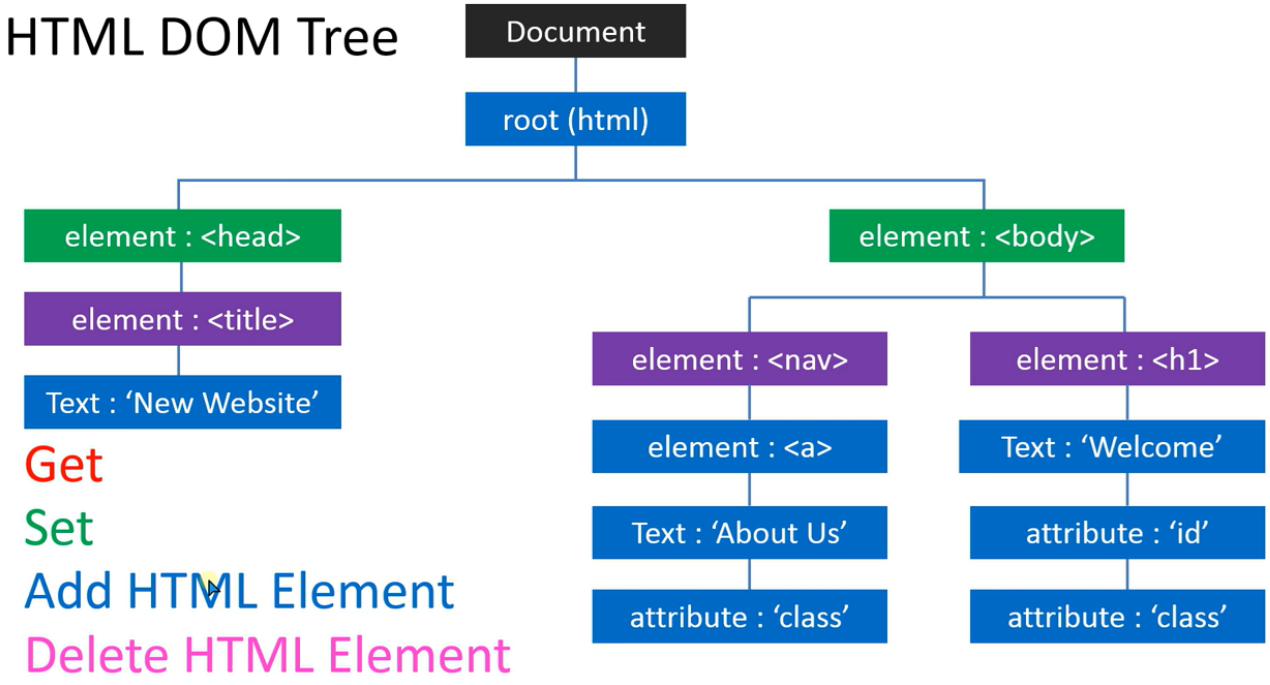
Selects a part of an element, such as the first letter or line.

```
p::first-line {
  color: green;
}

h1::before {
  content: "👉 ";
}
```

AJAX

```
<html>
<head>
    <body>
        <p id="demo">Here we load data .</p>
        <button onclick="loadData()">Click</button>

        <script>
            function loadData() {
                var xhttp = new XMLHttpRequest();
                xhttp.onreadystatechange = function(){
                    if(this.readyState == 4 && this.status == 200){
                        document.getElementById("demo").innerHTML =  (this.responseText);
                    }else {
                        document.getElementById("demo").innerHTML =  "error occured";
                    }
                }

                xhttp.open('GET' , "content/readme.txt", true);
                xhttp.send();

            }

        </script>

    </body>

</html>
```

DOM Manipulation



Targeting DOM Elements

- document
- document.all
- document.documentElement
- document.head
- document.title
- document.body
- document.images
- document.anchors
- document.links
- document.forms
- document.doctype
- document.URL
- document.baseURI
- document.domain

- Id → document.getElementById(id)
- Class Name → document.getElementsByClassName(name)
- Tag Name → document.getElementsByTagName(name)

DOM Get Methods

- innerText
- innerHTML
- getAttribute : give you value of attributes like id's name, style's css
- getAttributeNode: give you whole node id="ahmed", styles="border:1px"
    - to get the value use elem.getAttributeNode("style").value or simply use getAttribute
- attributes: gives all attributes of that element in object

DOM Set Methods

- innerText: will print as is, <button>meow</button > will be shown like this
- innerHTML: will generate the html result: meow
- setAttribute(name,newvalue): you pass the name of attribute and the newvalue to overwrite
- attributes: You get all attributes as array and then can change each one like this:
    - elem.attributes[2].value = "xyz"
- removeAttribute(*name*): deletes the *name* attribute from the element.

Query Selector

- querySelector(cssSelector): you pass cssSelector and it selects 1st element with that cssSelector
- querySelectorAll(cssSelector): returns an array with elements having same cssSelector. After wards you can select 1 element and do all the things above

DOM CSS Styling Methods

- style
    - console.log(elem.style.color)
    - elem.style.color = "green"
    - if css attribute is like this background-color. Remove hifen and capitalize next character. Eg: elem.style.backgroundColor = "green"
- className: make classes with predined styling and just change them
    - elem.className = "clicked-link"
    - elem.className = "clicked-link hovered" (you can add multiple classes)
- classList
    - This returns all the classes in array

- You can append classes: elem.classList.add(“xyz”) or elem.classList.add(“xyz abc”)
- You can remove classes: elem.classList.add(“xyz”) or elem.classList.add(“xyz abc”)

Assigning Events using HTML DOM

- Events: onclick, onmouseenter, onmouseleave
    - Example: document.getElementById("my-btn").onclick = functoinName
- EventListener: click, mouseenter
    - Example: document.getElementById("my-btn").addEventListener("click", funcName)
    - Example: document.getElementById("my-btn").addEventListener("mouseenter", function(){
        Console.log("meow")
      })

DOM Traversal Methods

- elem.parentNode: gives parent node of current node
- elem.childNodes: gives array of child nodes
- elem.firstChild and elem.lastChild: returns first and last child node
- elem.previousElementSibling: returns element before the current elem
- elem.nextElementSibling: returns element after the current elem

DOM Create Methods

- document.createElement("tagName"): creates an html tag element
- document.createTextNode("text node"): creates an html tag element

DOM Child Node Operations

- elem.appendChild(newNode): you pass the node here you make like above. This appends the elements in the last of all children
- elem.insertBefore(node1, node2): puts the node1 before node2
    - elem.insertBefore(node, elem.childNodes[0]): puts node as first child
- elem.removeChild(node): removes the child node
    - elem.removeChild(elem.childNodes[0]): removes first child
- elem.replaceChild(oldNode,newNode):
    - elem.replaceChild(elem.childNodes[0], elem.childNodes[elem.childNodes.length]): replaces first with last child

EXPRESS JS

Initial code:

```
cosnt express = require("express")
const app = express()

app.listen(3000, ()=>{
	console.log("connected on localhost:300")
})
```

HTTP Request types: read, create, update, delete

Express methods: get, post, put, delete, set("view engine", "ejs")

Route generic code:

```
app.method('route', (req, res) => {
})
```

Get Method to show a message on /:

```
app.get('/', (req, res) => {
	res.send('<h1>Welcome To Home Page</h1>')
})
```

Pass Value through params

```
app.get('/user/:userId', (req, res) => {
	id = req.params.userId
	res.send('<h1>Welcome To User number ' + id + '</h1>')
})
```

Pass values through query string

eg user enters this on url: localhost:300/search?name=meow&age=1&city=milkyway

```
app.get('/search', (req, res) => {
	const name = req.query.name
	const age = req.query.age
	const city = req.query.city
	res.send(`We found your kitty ${name} aged ${age} in ${city}`)
})
```

ExpressJS Response Method

- res.send(): Text, HTML, Object, Arrays, Buffers
- res.json(): JSON object
- res.redirect(): redirect to another page. you pass the route or address in function
- res.render(): Open HTML file using template engine like ejs
- res.download(): PDF, Image, Music, Video ...

ExpressJS Request Properties and Methods

- req.params
- req.query
- req.body: json data, form data
  - to accept json data: app.use(express.json())
  - to accept form data: app.use(express.urlencoded({extended: false}))
- req.cookies

EJS

EJS Template Tag Syntax

<% %> Control flow, no output

Example:

```
<ul>
   <% for(let i=1; i<=3; i++) { %>
       <li> Item <% i %> <li>
   <% } %>
</ul>
```

<%= %> Same as above. Ouputs escaped values (safe), prevents XSS attacks

Passing value in ejs template

```
app.get('/about', (req,res)=>{
   var name = 'meow'
   res.render("about", name)
})

//ejs
Hello <%= name %>
```

Passing multiple values in ejs template

```
app.get('/about', (req,res)=>{
   var name = "meow"
   var city = "milkyway"
   user = {name:name, city:city}

   //pass user as an object
   res.render("about", {user})
})

//ejs
Hello <%= user.name %> from <%= city =>
```

Passing array in ejs template

```
app.get('/about', (req,res)=>{
   var users = [
       {name: "Meow", city "milkyway"}
       {name: "woof", city "pluto"}
   ]

   //pass users as an object
   res.render("about", {users})
})

//ejs
<table>
<% users.forEach( user=> { %>
<tr>
  <td><%= user.name %></td>
  <td><%= user.city %></td>
</tr>
<% }) %>
</table>
```

Using forms in ejs

```
app.use(express.urlencoded({extended:false}))
app.post('/submit', (req,res)=>{
      const name = req.body.myname
      const city= req.body.mycity

      const message = `Hello, ${name} from ${city}`
      res.send(message)
})

//ejs
<form action="/submit" method="POST">
   Enter Name: <input type="text" name="myname">
   Enter city: <input type="text" name="mycity">

   <input type="submit">
</form>
```

Using forms in ejs and showing the form data on same page

```
app.use(express.urlencoded({extended:false}))
app.post('/submit', (req,res)=>{
      const name = req.body.myname
      const city= req.body.mycity

      const message = `Hello, ${name} from ${city}`
      res.render('form', {message:message})
})

app.get('/submit', (req, res)=>{
      res.render('form', {message: null})
})

//ejs
<form action="/submit" method="POST">
   Enter Name: <input type="text" name="myname">
   Enter city: <input type="text" name="mycity">

   <input type="submit">
</form>
<% if (message) { %>
  <p><%= message %></p>
<% if %>
```

EJS Template Partials

You can make different files and reuse them in different ejs files. Like components in React

```
<%- include('filename') -%>
```

Using Statics

Set middleware in your index.js

```
app.use(express.static('public'))
```

Remaing:

15 Middleware

18 Session

19 Authentiation

20 cookies

21 CSRF token