

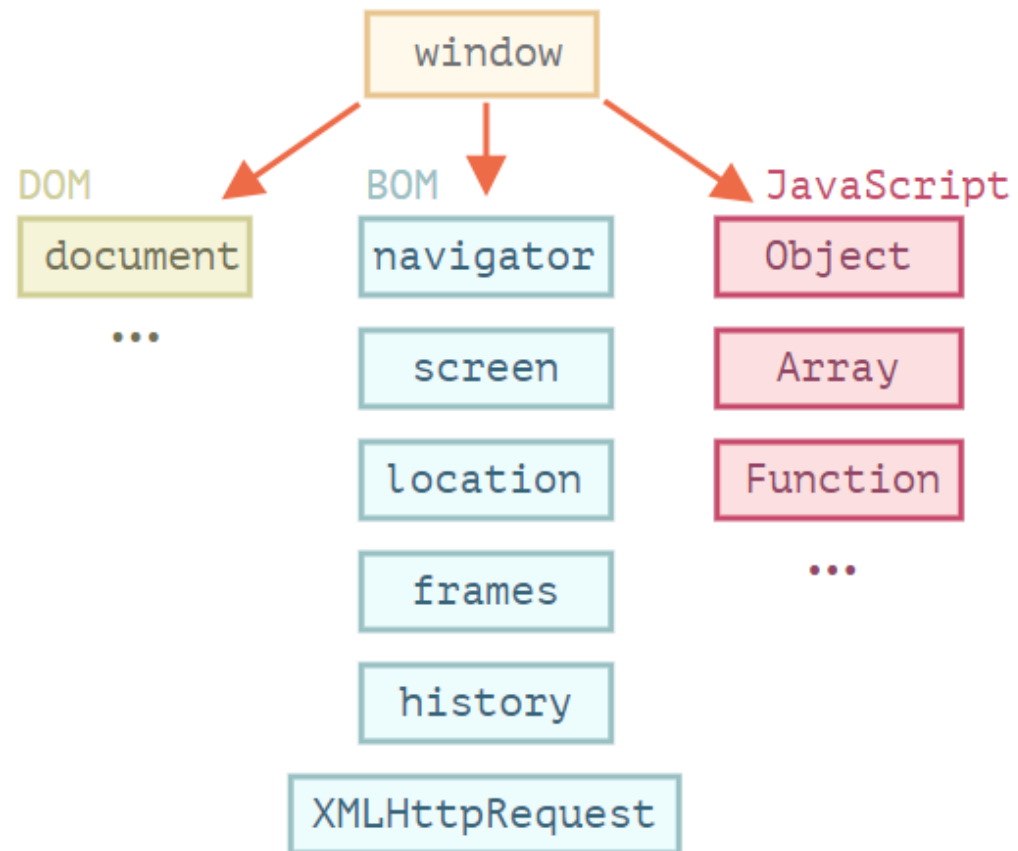


TECHIN

JavaScript DOM

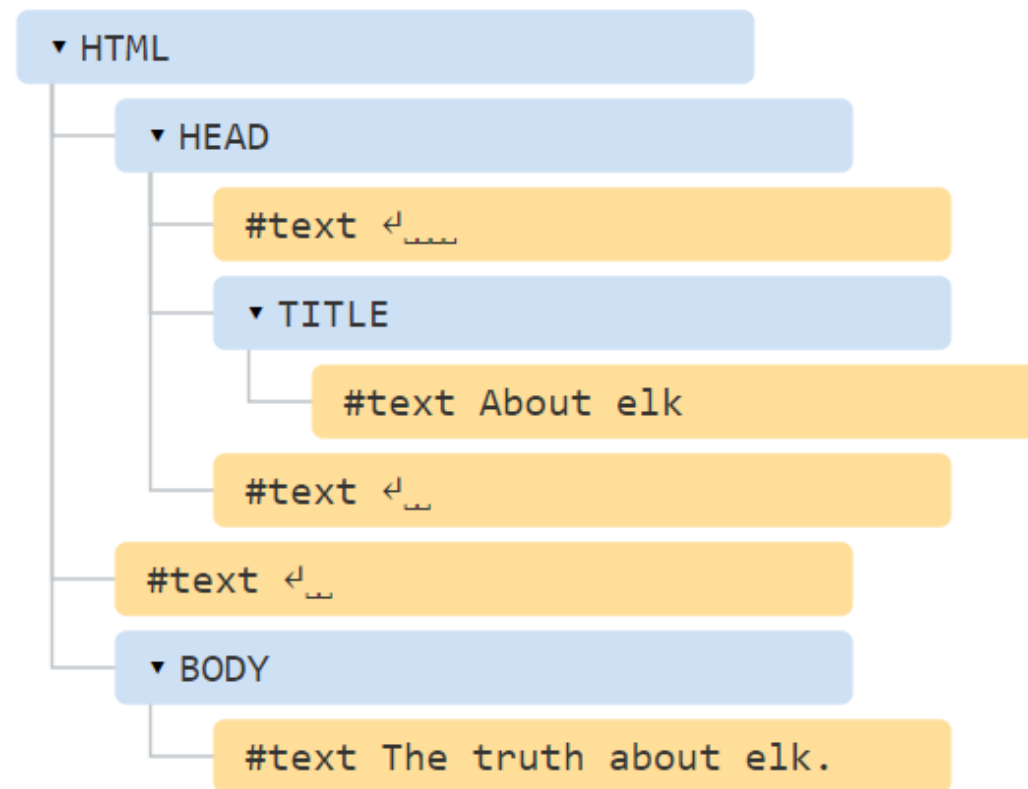
JavaScript DOM (Document Object Model)

Kai JS kodą paleidžiame naršyklėje:
window - root objektas, t. y. pagrindinis
objektas



JavaScript DOM

```
1 <!DOCTYPE HTML>
2 <html>
3 <head>
4   <title>About elk</title>
5 </head>
6 <body>
7   The truth about elk.
8 </body>
9 </html>
```



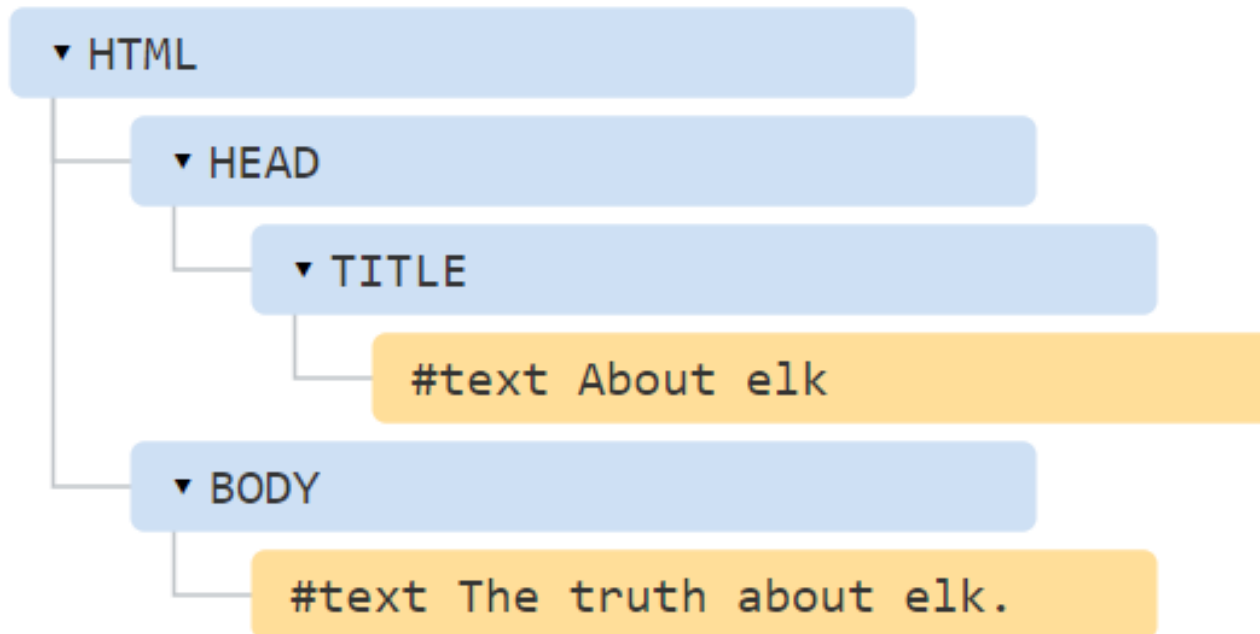
HTML elementai, jų tekstai, tarpai ir perėjimai į kitą eilutę (tarp elementų), sudaro DOM medį. Elementai vadinami mazgais (angl. DOM nodes).



JavaScript DOM

Jei HTML parašysime be tarpų:

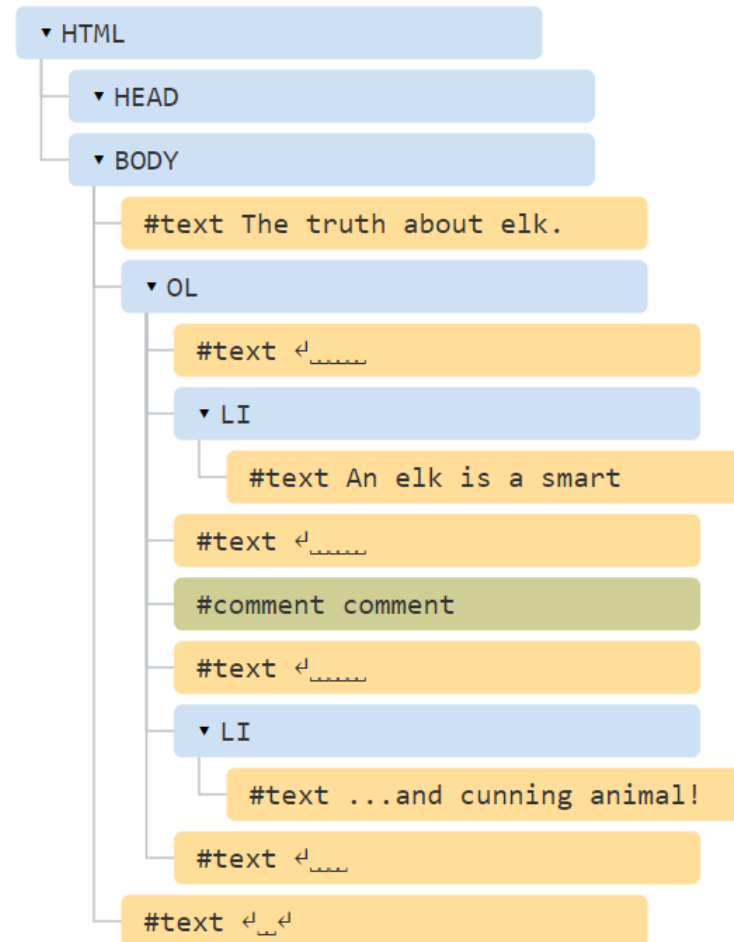
```
1 <!DOCTYPE HTML>  
2 <html><head><title>About elk</title></head><body>The truth about elk.</body></html>
```



JavaScript DOM

Komentarai taip pat DOM elementai.

```
1 <!DOCTYPE HTML>
2 <html>
3 <body>
4   The truth about elk.
5   <ol>
6     <li>An elk is a smart</li>
7     <!-- comment -->
8     <li>...and cunning animal!</li>
9   </ol>
10 </body>
11 </html>
```



JavaScript DOM

**Dažniausiai dirbame
su:**

- **document - pagrindiniu DOM mazgu.**
- **element nodes - HTML elementais.**
- **text nodes - tekstas, enter ir tarpai.**
- **Kartais dar su komentarais.**



JavaScript DOM

**Norint dirbti su HTML elementais,
nustatyti ar pakeisti jų parametrus,
juos reikia surasti:**

**HTML elementų radimas pagal jų HTML
id:**

```
1 <div id="elem">
2   <div id="elem-content">Element</div>
3 </div>
4
5 <script>
6   // get the element
7   let elem = document.getElementById('elem');
8
9   // make its background red
10  elem.style.background = 'red';
11 </script>
```



JavaScript DOM

HTML elementų radimas pagal elemento pavadinimą:

```
<!DOCTYPE html>
<html>
<body>

<h2>Finding HTML Elements by Tag Name</h2>

<div id="main">
  <p>The DOM is very useful.</p>
  <p>This example demonstrates the <b>getElementsByName</b> method.</p>
</div>

<p id="demo"></p>

<script>
var x = document.getElementById("main");
var y = x.getElementsByTagName("p");
document.getElementById("demo").innerHTML =
'The first paragraph (index 0) inside "main" is: ' + y[0].innerHTML;
</script>

</body>
</html>
```

Kadangi elementų gali būti keli, grąžinama elementų kolekcija ir norėdami rasti konkretų elementą nurodome jo indeksą (kaip masyve).

Jei HTML'e pasikeičia elementų kiekis, rezultatas atsinaujina.



JavaScript DOM

HTML elementų radimas pagal **klasės pavadinimą**:

```
<!DOCTYPE html>
<html>
<body>

<h2>Finding HTML Elements by Class Name</h2>

<p>Hello World!</p>

<p class="intro">The DOM is very useful.</p>
<p class="intro">This example demonstrates the <b>getElementsByClassName</b> method.</p>

<p id="demo"></p>

<script>
var x = document.getElementsByClassName("intro");
document.getElementById("demo").innerHTML =
'The first paragraph (index 0) with class="intro": ' + x[0].innerHTML;
</script>

</body>
</html>
```

Kadangi elementų su vienoda klase gali būti keli, grąžinama elementų kolekcija ir konkretų elementą nurodome su indeksu.

Jei pasikeičia elementų kiekis, rezultatas atsinaujina.



JavaScript DOM

HTML elementų radimas pagal **css** **selektorių:**

```
1 <ul>
2   <li>The</li>
3   <li>test</li>
4 </ul>
5 <ul>
6   <li>has</li>
7   <li>passed</li>
8 </ul>
9 <script>
10  let elements = document.querySelectorAll('ul > li:last-child');
11
12  for (let elem of elements) {
13    alert(elem.innerHTML); // "test", "passed"
14  }
15 </script>
```

Grąžina kolekciją visų elementų, atitinkančių duotą css selektorių.

Jei elementų skaičius keičiasi, rezultatas neatsinaujina.
Jei reikia pirmo, atitinkančio css selektorių,
naudojame `querySelector`.



JavaScript DOM elementų turinio keitimas

```
<html>
<body>

<p id="p1">Hello World!</p>

<script>
document.getElementById("p1").innerHTML = "New text!";
</script>

</body>
</html>
```

Perrašo visą buvusį elemento turinį.

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript can Change HTML</h2>

<p id="p1">Hello World!</p>

<script>
document.getElementById("p1").innerHTML += "New text!";
</script>

<p>The paragraph above was changed by a script.</p>

</body>
</html>
```

Papildo elemento turinį.



JavaScript DOM elementų atributai

```
const div = document.createElement('div');  
// create a new div referenced in the variable 'div'
```

```
div.setAttribute('id', 'theDiv');  
// if id exists update it to 'theDiv' else create an id  
// with value "theDiv"
```

```
div.getAttribute('id');  
// returns value of specified attribute, in this case  
// "theDiv"
```

```
div.removeAttribute('id');  
// removes specified attribute
```



JavaScript DOM elementų atributų pridėjimas

```
1 <button type="button" id="myBtn">Click Me</button>
2
3 <script>
4     // Selecting the element
5     var btn = document.getElementById("myBtn");
6
7     // Setting new attributes
8     btn.setAttribute("class", "click-btn");
9     btn.setAttribute("disabled", "");
10 </script>
```



JavaScript DOM elementų atributų gavimas



```
1  <a href="https://www.google.com/" target="_blank" id="myLink">Google</a>
2
3  <script>
4      // Selecting the element by ID attribute
5      var link = document.getElementById("myLink");
6
7      // Getting the attributes values
8      var href = link.getAttribute("href");
9      alert(href); // Outputs: https://www.google.com/
10
11     var target = link.getAttribute("target");
12     alert(target); // Outputs: _blank
13 </script>
```



JavaScript DOM elementų atributų keitimas

```
1  <a href="#" id="myLink">Tutorial Republic</a>
2
3  <script>
4      // Selecting the element
5      var link = document.getElementById("myLink");
6
7      // Changing the href attribute value
8      link.setAttribute("href", "https://www.tutorialrepublic.com");
9  </script>
```



JavaScript DOM elementų atributų šalinimas

```
1  <a href="https://www.google.com/" id="myLink">Google</a>
2
3  <script>
4      // Selecting the element
5      var link = document.getElementById("myLink");
6
7      // Removing the href attribute
8      link.removeAttribute("href");
9  </script>
```



JavaScript DOM elementų atributų keitimas

```
<!DOCTYPE html>
<html>
<body>



<script>
document.getElementById("myImage").src = "landscape.jpg";
</script>

</body>
</html>
```



JavaScript DOM elementų stiliaus savybių (css) keitimas

```
const div = document.createElement('div');  
// create a new div referenced in the variable 'div'
```

```
div.style.color = 'blue';  
// adds the indicated style rule
```

```
div.style.cssText = 'color: blue; background: white';  
// adds several style rules
```

```
div.setAttribute('style', 'color: blue; background: white');  
// adds several style rules
```



JavaScript DOM elementų stiliaus savybių (css) keitimas

```
<html>
<body>

<p id="p2">Hello World!</p>

<script>
document.getElementById("p2").style.color = "blue";
</script>

<p>The paragraph above was changed by a script.</p>

</body>
</html>
```

Atkreipkite dėmesį, kad JS kalboje kitaip užrašomos css savybės: nenaudojant brūkšnio - , o sujungiant žodžius ir parašant juos camelCase. Pvz. `backgroundColor` (css: `background-color`). - nenaudojame, nes tai atimties ženklas.

Visas sąrašas: https://www.w3schools.com/jsref/dom_obj_style.asp



JavaScript DOM elementų stiliaus savybių (css) keitimas paspaudus pelytę

```
<!DOCTYPE html>
<html>
<body>

<h1 id="id1">My Heading 1</h1>

<button type="button"
onclick="document.getElementById('id1').style.color = 'red'">
Click Me!</button>

</body>
</html>
```



JavaScript DOM elementų klasės keitimas - perrašymas

.className parodo, kokia elemento klasė. Galima šią savybę keisti.

```
1 <body class="main page">
2   <script>
3     alert(document.body.className); // main page
4   </script>
5 </body>
```

```
1 <!DOCTYPE html>
2 <html>
3
4 <body>
5
6   <h2>JavaScript can Change HTML</h2>
7
8   <p id="p1" class="hel">Hello World!</p>
9
10  <p>The paragraph above was changed by a script.</p>
11
12  <script>
13    let elem = document.getElementById("p1");
14    console.log(elem.className); //išvedam į konsolę klasės vardą
15    elem.className = "bell"; //pakeičiam, perrašom klasės vardą
16    console.log(elem.className); //išvedam naują klasės vardą
17  </script>
18 </body>
19 </html>
```



JavaScript DOM elementų klasės keitimas - pridėjimas ir kt.

`classList` tai elemento klasių sąrašas (nes elementas gali turėti ne vieną klasę)

```
div.classList.add('new');  
// adds class "new" to your new div  
  
div.classList.remove('new');  
// remove "new" class from div  
  
div.classList.toggle('active');  
// if div doesn't have class "active" then add it, or if  
// it does, then remove it
```



JavaScript DOM elementų klasės keitimas - pridėjimas ir kt.

classList tai elemento klasių sąrašas (nes elementas gali turėti ne vieną klasę)

```
1 <!DOCTYPE html>
2 <html>
3
4 <body>
5
6   <h2>JavaScript can Change HTML</h2>
7
8   <p id="p1" class="hell">Hello World!</p>
9
10  <p>The paragraph above was changed by a script.</p>
11
12  <script>
13    let elem = document.getElementById("p1");
14    elem.classList.add("bell"); //prideda naują klasę
15
16    console.log(elem.classList[0]); //parodo elemento klasių sąrašo 1 klasę t.y. hell
17  </script>
18 </body>
19 </html>
```



JavaScript DOM įvykiai

JavaScript gali reaguoti į įvairius **DOM elementų įvykius** ir vykdyti komandas, t.y. atlikti tam tikrus veiksmus.

Įvykiai:

- Pelės įvykiai (`click`, `dblclick`, `mousemove`, `mouseover`, `mousewheel`);
- Lietimo įvykiai (planšetės, telefonai ir kt.) (`touchstart`, `touchmove`);
- Klaviatūros įvykiai (`keypress`, `keyup`);
- Formos įvykiai (`focus`, `change`, `submit`);
- Window įvykiai (`scroll`, `resize`);



JavaScript DOM **ıvykis** **onclick**

```
<!DOCTYPE html>
<html>
<body>

<p>Click "Try it" to execute the displayDate() function.</p>

<button id="myBtn">Try it</button>

<p id="demo"></p>

<script>
document.getElementById("myBtn").onclick = displayDate;

function displayDate() {
    document.getElementById("demo").innerHTML = Date();
}
</script>

</body>
</html>
```



JavaScript DOM įvykis **onload**

```
<!DOCTYPE html>
<html>
<body onload="checkCookies()">

<p id="demo"></p>

<script>
function checkCookies() {
  var text = "";
  if (navigator.cookieEnabled == true) {
    text = "Cookies are enabled.";
  } else {
    text = "Cookies are not enabled.";
  }
  document.getElementById("demo").innerHTML = text;
}
</script>

</body>
</html>
```

**Objektas
navigator
saugo informaciją
apie naršyklę.**

**Parašytas kodas
patikrina, ar
cookies (liet.
slapukai)
naršyklėje
aktyvuoti.**



JavaScript DOM įvykis

onchange

```
<!DOCTYPE html>
<html>
<head>
<script>
function myFunction() {
  var x = document.getElementById("fname");
  x.value = x.value.toUpperCase();
}
</script>
</head>
<body>
```

Enter your name: `<input type="text" id="fname" onchange="myFunction()">`

`<p>`When you leave the input field, a function is triggered which transforms the input text to upper case.`</p>`

```
</body>
</html>
```

Atkreipkite dėmesį, kaip paimama formos tekstinio lauko reikšmė **`document.getElementById("fname").value`**



JavaScript DOM įvykiai **onmouseover**, **onmouseout** (užvedus, nuvedus pelytę)

```
<!DOCTYPE html>
<html>
<body>

<div onmouseover="mOver(this)" onmouseout="mOut(this)"
style="background-color:#D94A38;width:120px;height:20px;padding:40px;">
Mouse Over Me</div>

<script>
function mOver(obj) {
  obj.innerHTML = "Thank You"
}

function mOut(obj) {
  obj.innerHTML = "Mouse Over Me"
}
</script>

</body>
</html>
```

Žodis „this“ simbolizuoja esamą objektą – šiuo atveju, tai šis div'as.

innerHTML nuo innerText skiriasi tuo, jog naudojant innerHTML, galime įterpti HTML kodą.

Tarkime, su innerHTML, kodas **Hello** tekstą ir patamsins. Jei naudotume innerText, tekstą taip tiesiai ir įterptų – netraktuotų kaip HTML kodą.



JavaScript DOM įvykis

onfocus

(kai į formos teksto įvedimo lauką padedama pelė)

```
<!DOCTYPE html>
<html>
<head>
<script>
function myFunction(x) {
  x.style.background = "yellow";
}
</script>
</head>
<body>
```

Enter your name: <input type="text" onfocus="myFunction(this)">

<p>When the input field gets focus, a function is triggered which changes the background-color.</p>

```
</body>
</html>
```



JavaScript DOM įvykis **onblur**

(kai formos teksto įvedimo laukas praranda fokusą)

```
<!DOCTYPE html>
<html>
<body>
<h1>HTML DOM Events</h1>
<h2>The blur Event</h2>
```

Enter your name:

`<p>`When you leave the input field, a function is triggered which transforms the input text to upper case.`</p>`

```
<script>
function myFunction() {
  let x = document.getElementById("fname");
  x.value = x.value.toUpperCase();
}
</script>

</body>
</html>
```

Šiame pavyzdyje, žodelis „this“ nėra naudojamas.

HTML elementas
JavaScript kode randamas
„getElementById“ pagalba
– tai taip pat priimtinas
būdas.



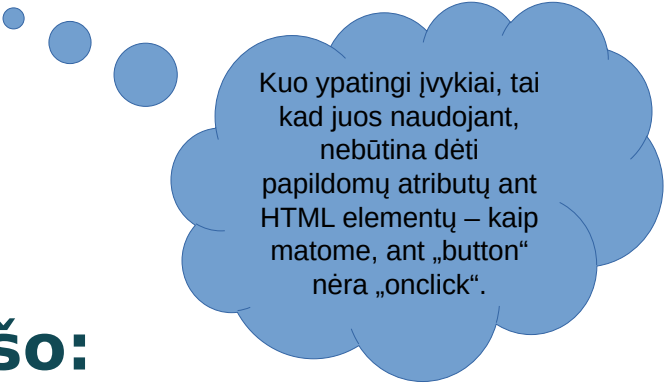
JavaScript DOM įvykių klausymas

Bet kuriam DOM elementui galima parašyti įvykių klausymo metodą. Tai šiuolaikinis būdas dirbti su įvykiais.

```
<button id="myBtn">Try it</button>
```

```
<script>  
document.getElementById("myBtn").addEventListener("click", myFunction);
```

```
function myFunction() {  
    alert ("Hello World!");  
}  
</script>
```



Kuo ypatingi įvykiai, tai kad juos naudojant, nebūtina dėti papildomų atributų ant HTML elementų – kaip matome, ant „button“ nėra „onclick“.

Įvykis gali būti bet kuris iš sąrašo:

https://www.w3schools.com/jsref/dom_obj_event.asp



JavaScript DOM įvykių savybės ir metodai

DOM įvykiai(events) gali turėti savybes ir metodus.

https://www.w3schools.com/jsref/dom_obj_event.asp

```
<!DOCTYPE html>
<html>
<body>

<h2 onclick="showCoords(event)">Click this heading to get the x
(horizontal) and y (vertical) coordinates of the mouse pointer when it
was clicked.</h2>

<p><strong>Tip:</strong> Try to click different places in the heading.
</p>

<p id="demo"></p>

<script>
function showCoords(event) {
  var x = event.clientX;
  var y = event.clientY;
  var coords = "X coords: " + x + ", Y coords: " + y;
  document.getElementById("demo").innerHTML = coords;
}
</script>

</body>
</html>
```

Turi vadintis „event“, ne kaip kitaip. „this“ yra skirtingas dalykas.

Atkreipkite dėmesį, kad norint dirbti su įvykių savybėmis ar metodais, įvykis turi būti perduodamas funkcijai



JavaScript DOM įvykių savybės ir metodai, moderniau

```
<!DOCTYPE html>
<html>
  <body>
    <h2 id="heading">
      Click this heading to get the x (horizontal) and y (vertical) coordinates
      of the mouse pointer when it was clicked.
    </h2>

    <p><strong>Tip:</strong> Try to click different places in the heading.</p>

    <p id="demo"></p>

    <script>
      function showCoords(event) {
        const x = event.clientX;
        const y = event.clientY;

        const coords = "X coords: " + x + ", Y coords: " + y;
        document.getElementById("demo").innerHTML = coords;
      }

      const heading = document.getElementById("heading");
      heading.addEventListener("click", showCoords);
    </script>
  </body>
</html>
```

JavaScript DOM įvykių savybės ir metodai

Vienas iš svarbiausių metodų yra **event.target**, jo pagalba skriptas mato HTML elementą, ant kurio vyksta įvykis, pvz. paspaudimas.

JS script.js > ...

```
1 document.body.addEventListener("click", (event) => {  
2   event.preventDefault();  
3   console.log(event.target);  
4   event.target.style.color = "red";  
5 });
```

Standartinio naršyklės elgesio įvykus įvykiui - persikrovimo, stabdymas.

Įvykio perdavimas callback funkcijai

```
10 <body>  
11   <h2>HTML Forms</h2>  
12  
13   <form action="#">  
14     <label for="fname">First name:</label><br>  
15     <input type="text" id="fname" name="fname" value="John"><br>  
16     <label for="lname">Last name:</label><br>  
17     <input type="text" id="lname" name="lname" value="Doe"><br><br>  
18     <input type="submit" value="Submit">  
19   </form>  
20  
21   <p>If you click the "Submit" button, the form-data will be sent to a page called "/action_page.php".</p>  
22  
23 </body>
```



JavaScript DOM įvykių savybės ir metodai

Metodo `matches()` pagalba patikrinsime, ar paspaustas elementas turi id "my-button"

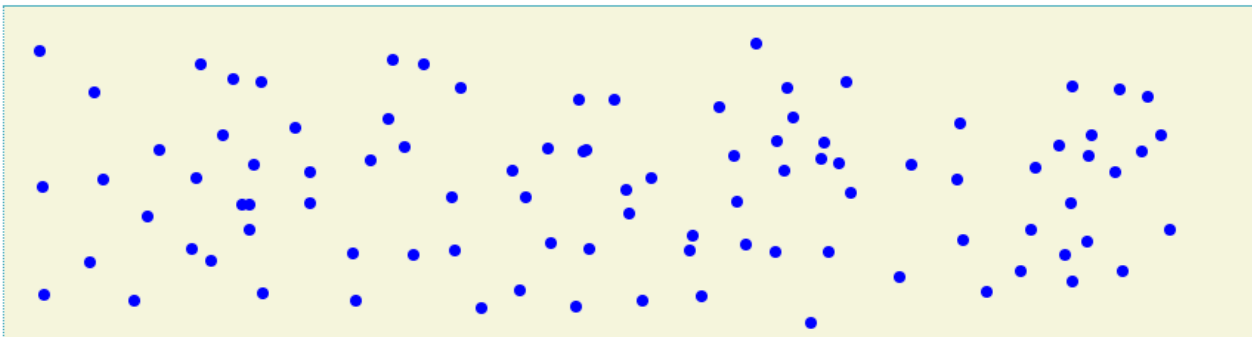
```
document.addEventListener("click", function(event) {  
    // Check if the event target is a specific element  
    if (event.target.matches("#my-button")) {  
        // Do something here  
        console.log("The #my-button element was clicked!");  
    }  
});
```



JavaScript DOM įvykių savybės ir metodai (pavyzdys)

```
1 <style>
2   body {
3     height: 200px;
4     background: beige;
5   }
6   .dot {
7     height: 8px; width: 8px;
8     border-radius: 4px; /* rounds corners */
9     background: blue;
10    position: absolute;
11  }
12 </style>
13 <script>
14   window.addEventListener("click", event => {
15     let dot = document.createElement("div");
16     dot.className = "dot";
17     dot.style.left = (event.pageX - 4) + "px";
18     dot.style.top = (event.pageY - 4) + "px";
19     document.body.appendChild(dot);
20   });
21 </script>
```

Įvykio perdavimas
callback funkcijai



JavaScript DOM įvykių savybės ir metodai

Panagrinėti:

https://www.w3schools.com/jsref/event_target.asp

https://www.w3schools.com/jsref/event_preventdefault.asp

https://www.w3schools.com/jsref/event_pageex.asp



JavaScript DOM elementų kūrimas

1. Teksto kūrimas:

```
var node = document.createTextNode("This is new.");
```

2. HTML elemento kūrimas:

```
var para = document.createElement("p");
```

3. Teksto ir elemento sujungimas

(analogiškai vyksta ir dviejų elementų sujungimas, naujas elementas pridedamas tėvinio elemento gale)

```
para.appendChild(node);
```

prie kurio elemento

kurį elementą jungiame



JavaScript DOM elementų kūrimas

```
<div id="div1">  
<p id="p1">This is a paragraph.</p>  
<p id="p2">This is another paragraph.</p>  
</div>
```

```
<script>  
var para = document.createElement("p");  
var node = document.createTextNode("This is new.");  
para.appendChild(node);  
var element = document.getElementById("div1");  
element.appendChild(para);  
</script>
```

!!! Naujai sukurtas elementas nematomas puslapyje, kol jis nepridedamas prie jau esamo elemento (appendChild)

This is a paragraph.

This is another paragraph.

This is new.



JavaScript DOM elementų kūrimas

Įterpiant naują elementą, galima nurodyti jo vietą:

```
element.insertBefore(para, child);
```

Į kurį
element
ą
įterpia
m

Kurį
element
ą
įterpiam

Prieš
kurį
element
ą
įterpiam



JavaScript DOM elementu kūrimas

```
<div id="div1">  
<p id="p1">This is a paragraph.</p>  
<p id="p2">This is another paragraph.</p>  
</div>
```

```
<script>  
var para = document.createElement("p");  
var node = document.createTextNode("This is new.");  
para.appendChild(node);
```

```
var element = document.getElementById("div1");  
var child = document.getElementById("p1");  
element.insertBefore(para, child);  
</script>
```



JavaScript DOM elementų šalinimas

```
<div>  
<p id="p1">This is a paragraph.</p>  
<p id="p2">This is another paragraph.</p>  
</div>
```

```
<button onclick="myFunction()">Remove Element</button>
```

```
<script>  
function myFunction() {  
    var elmnt = document.getElementById("p1");  
    elmnt.remove();  
}  
</script>
```



JavaScript DOM elementų pakeitimas

```
<div id="div1">  
<p id="p1">This is a paragraph.</p>  
<p id="p2">This is another paragraph.</p>  
</div>
```

```
<script>  
var parent = document.getElementById("div1");  
var child = document.getElementById("p1");  
var para = document.createElement("p");  
var node = document.createTextNode("This is new.");  
para.appendChild(node);  
parent.replaceChild(para, child);  
</script>
```

Kuriame elem. vyks
pokyčiai

Kuo keisim
(naujas elem.)

Kurį elem. Keisim
(senas elem.)



JavaScript DOM pavyzdys

```
<!-- your html file: -->
<body>
  <h1>
    THE TITLE OF YOUR WEBPAGE
  </h1>
  <div id="container"></div>
</body>
```

```
// your javascript file
const container = document.querySelector('#container');


const content = document.createElement('div');
content.classList.add('content');
content.textContent = 'This is the glorious text-content!';


container.appendChild(content);
```



Navigavimas DOM medžiu

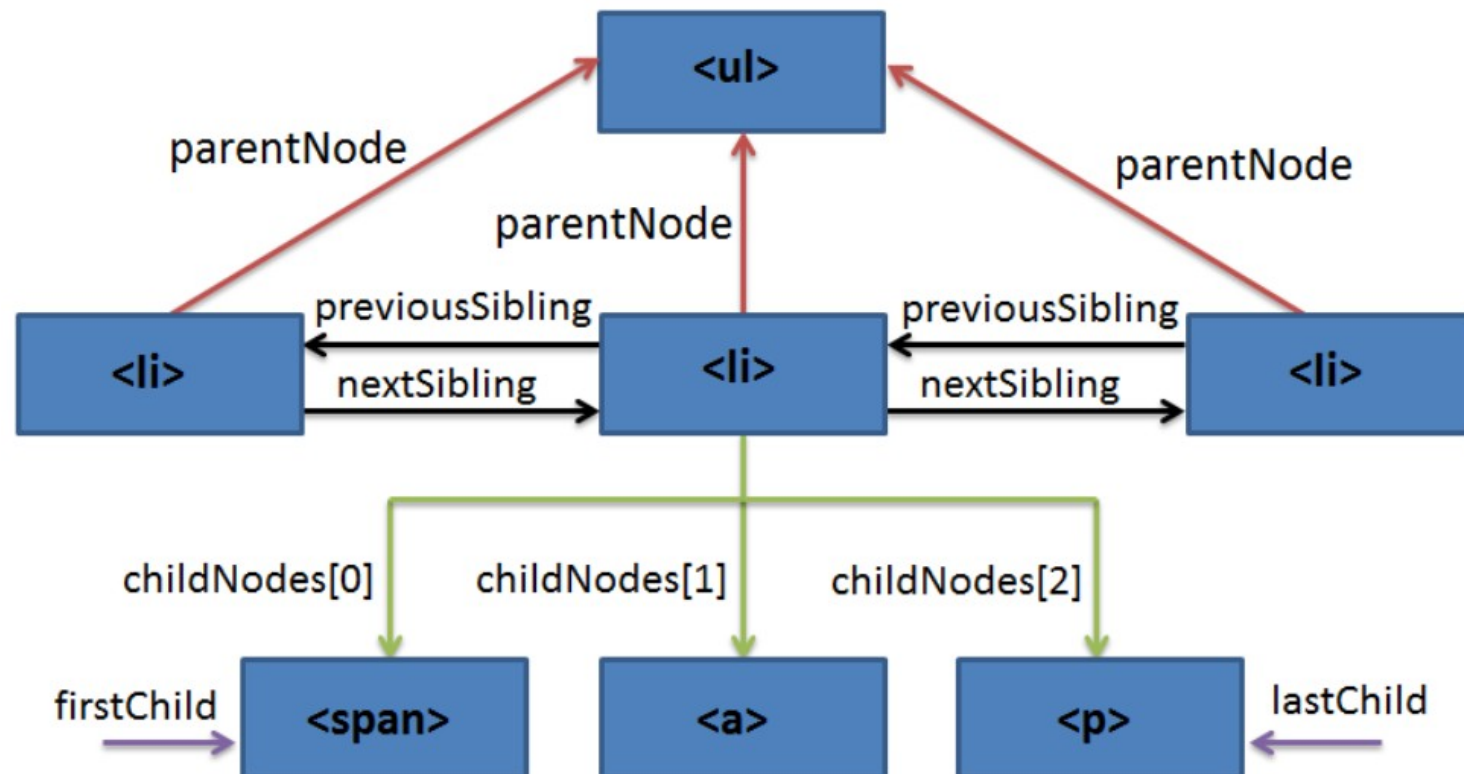
```
<ul>
<li>node</li>
<li><span>node</span><a href="#">node</a><p>node</p></li>
<li>node</li>
</ul>
```

 **nextElementSibling**

 **nextSibling**

Imami tik html elementai

Imami visi elementai ir
tekstai ir enter



Navigavimas DOM medžiu pavyzdys

```
1 <div id="main">
2   <h1 id="title">My Heading</h1>
3   <p id="hint"><span>This is some text.</span></p><hr>
4 </div>
```

```
5
6 <script>
7   var title = document.getElementById("title");
8   alert(title.previousSibling.nodeName); // Outputs: #text
9
10  var hint = document.getElementById("hint");
11  alert(hint.nextSibling.nodeName); // Outputs: HR
12 </script>
```

nodeName yra read-only savybė, grąžinanti node vardą kaip string.



JavaScript darbas su formomis

```
<form action="/signup" method="post" id="signup">  
</form>
```

Formos elementas **HTMLFormElement** turi du svarbiausius atributus:

action - nurodo URL, kuris bus užkrautas kai patvirtinsime formą, t.y. nuspausime mygtuką submit, tai gali būti failas su skriptu formos duomenims apdoroti.

method - nurodo HTTP metodą, kuriuo bus patvirtinta forma (GET, POST...)



JavaScript darbas su formomis

```
<form action ="/signup" method = "post" id="signup" name="signup"></form>
```

Formą galime pasiekti per ID ir per vardą:

```
const form = document.getElementById('signup');
```

document.forms grąžina kolekciją visų dokumente esančių formų

```
const form = document.forms['login'];
```



JavaScript darbas su formomis

Formos laukų išvalymas (reset):

```
<form id="form">  
  First name: <input type="text" name="firstname"><br>  
  Last name: <input type="text" name="lastname"><br><br>  
  <input type="button" onclick="resetForm()" value="Reset form">  
</form>
```

```
function resetForm() {  
  document.getElementById("form").reset();  
}
```



JavaScript darbas su formomis

Kai norime ką nors daryti pvz. atlikti formos duomenų validaciją po to, kai nuspaudžiame mygtuką submit, reikia pridėti formai įvykių klausymą ir kai norimas įvykis įvyksta iškviečiama funkcija.

```
const form = document.getElementById('signup');  
  
form.addEventListener('submit', (event) => {  
    // handle the form data  
});
```



JavaScript darbas su formomis

Kad paspaudus submit mygtuką puslapis nepersikrautų, reikia pridėti **event.preventDefault()** metodą.

```
form.addEventListener('submit', (event) => {  
    // stop form submission  
    event.preventDefault();  
});
```



JavaScript darbas su formomis

Formos elementų pasiekimas galimas su bet kuriuo iš DOM metodų:

`getElementsByName(),`
`getElementById(),`
`querySelector()`

```
const form = document.getElementById('signup');
```

arba:

```
form.elements[1]; // by index  
form.elements['email']; // by name  
form.elements['email']; // by id
```



JavaScript darbas su formomis

Formos elementų reikšmės pasiekiamos su savybe value:

```
const form = document.getElementById('signup');  
const name = form.elements['name'];  
const email = form.elements['email'];  
  
// getting the element's value  
let fullName = name.value;  
let emailAddress = email.value;
```

arba

```
var age = document.getElementById("age").value;
```



JavaScript darbas su formomis

Dar vienas būdas, kaip paimti į formos laukus įrašytus duomenis:

```
<!DOCTYPE html>
<html>
<head>
<script>
function validateForm() {
    var x = document.forms["myForm"]["fname"].value;
    if (x == "") {
        alert("Name must be filled out");
        return false;
    }
}
</script>
</head>
<body>

<form name="myForm" onsubmit="return validateForm()">
    Name: <input type="text" name="fname">
    <input type="submit" value="Submit">
</form>

</body>
</html>
```



JavaScript formos validacija

HTML:
L:

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <title>JavaScript Form Demo</title>
5      <meta name="viewport" content="width=device-width, initial-scale=1.0" />
6      <link rel="stylesheet" href="css/style.css" />
7  </head>
8  <body>
9      <div class="container">
10         <form action="signup.html" method="post" id="signup">
11             <h1>Sign Up</h1>
12             <div class="field">
13                 <label for="name">Name:</label>
14                 <input type="text" id="name" name="name" placeholder="Enter your fullname" />
15                 <small></small>
16             </div>
17             <div class="field">
18                 <label for="email">Email:</label>
19                 <input type="text" id="email" name="email" placeholder="Enter your email address" />
20                 <small></small>
21             </div>
22             <div class="field">
23                 <button type="submit" class="full">Subscribe</button>
24             </div>
25         </form>
26     </div>
27     <script src="js/app.js"></script>
28 </body>
29 </html>
```

<https://www.javascripttutorial.net/sample/dom/form/css/style.css>

<https://www.javascripttutorial.net/sample/dom/form/js/app.js>

Paaškinimai: <https://www.javascripttutorial.net/javascript-dom/javascript-form/>



The FormData object

Tai objektas leidžiantis lengvai pasiimti duomenis iš formos ir paversti objektu arba JSON

```
1 // sukuriame reference į DOM formą
2 let form = document.querySelector("#signup");
3
4 form.addEventListener("submit", (e) => {
5   e.preventDefault();
6
7   // paimami duomenys iš formos
8   let data = new FormData(form);
9
10  //sukuriamas tuščia js objektas
11  var object = {};
12
13  //iteruojant per FormData objektą, duomenys sudedami į sukurta objektą
14  data.forEach((value, key) => (object[key] = value));
15
16  //js objektas paverčiamas json
17  var json = JSON.stringify(object);
18
19  console.log(object);
20  console.log(json);
21 });
```

