# **DOM Events (įvykis)**

Dokumento objekto modelio registruojami įvesties įvykiai

## Events

HTML DOM leidžia Javascript reaguoti į HTML pasikeitimo ar įvesties įvykius. Ir atitinkamai javascript kodas gali būti paleistas nutikus registruotam įvykiui html medyje. Kaip pvz. naudotojas paspaudė pelės mygtuką ant kažkurio html elemento.

HTML elementai, net turi tam pritaikytas atributus, su kuriais galima paleisti Javascript kodą.

| <h1 onclick="alert('You just clicked')">Click on this text!</h1> |
| --- |

**Kiti HTML įvykių (events) pavyzdžiai:**

* Kuomet naudotojas paspaudė pelės mygtuką.
* Kuomet puslapis persikrauna.
* Kuomet puslapio paveiksliukai buvo užkrauti.
* Kuomet pelės kursorius užėjo arba nuėjo nuo html elemento.
* Kuomet įvesties laukas pasikeitė.
* Kuomet html forma buvo submit’tinta
* Kuomet naudotojas paspaudė klaviatūros mygtukus.

Event’us galime ir priskirti pačiam elementui, naudojant prieš tai išmoktus DOM metodus ir atributus.

| *element = document.querySelector('p.text'); element.onclick = function() {} // ant pauspaudimo paleis funkcija* |
| --- |

## Event onload ir onunload

Šie event’ai yra šiektiek svarbesni, nes mes galime indikuoti, kada atitinkamas komponentas yra pilnai užkrautas.

Jei puslapis krauna scriptus, kuriuos užkrauti reikia daugiau laiko, tai jūsų scripta gali užkrauti greičiau, arba kai kurie html elementai gali būti neužkrauti, nors jūsų Javascript užkrautas. Tokiu atvėju gali atsirasti taip vadinamos “race conditions” kuomet jūsų Javascript pradės ieškoti elementų su querySelector() bet jų neras, nes jie dar neegzistuos dom’e. Tokiu atveju galime klausytis onload event’o kuris pasileis, kada naršyklė užkraus visus puslapio resursus.

Pvz.

| window.onload = function() { // Run my code } |
| --- |

Visa DOM event elementų sąraša galite rasti šioje nuorodoje:

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

## Event listeners (įvykių klausytojai)

Kiekvienam DOM eventui galime pridėti vadinama įvykio klausytoja, event listener, tai yra specialus metodas, kurį turi kiekvienas HTML elemento Node’as. Šis metodas priima du argumentus, event’o pavadinima ir callBack funkcija, kuri bus pašaukta kada tas eventas įvyks.

| const button = document.querySelector(".button");  function handleClick() {  console.log("🐛 IT GOT CLICKED!!!"); }  button.addEventListener("click", handleClick); |
| --- |

Taip pat event listener’ius galime ištrinti, ypatingai, jei mums jo daugiau nereikia. Ypatingai tai svarbu, jei jūs prikabinote eventlistiner ir norite ištrinti html taga, tai tuomet gera praktika taikant, reikia prieš ištrintant html elementa ištrinti ir jo eventlistiner’ius jei jie buvo nustatyti, nes kitaip jie liks atmintyje, ir taip vadinamas atsiras “Memory Leak” atminties nutekėjimas.

| const button = document.querySelector(".button");  function handleClick() {  console.log("🐛 IT GOT CLICKED!!!"); }  button.addEventListener("click", handleClick);  butts.removeEventListener("click", handleClick); |
| --- |

Atkreipkite dėmesį, kad callback funkcija registruoju atskirai. Niekas netrukdo, ją paduoti kaip anonimine funkcija, tačiau jei ateityje planuojate šį event listiner’į ištrinti, su anonimine funkcija to nepavyks, tik jei deklaravotę ją atskirai ir padavėte kaip argumenta.

Taip pat galime vienu metu uždėti tą patį event listener daugiau nei vienam elementui.

| const allButtons = document.querySelectorAll(".button");  function handleClick(event) {  console.log(event.target); }  allButtons.forEach(btn => {  btn.addEventListener("click", handleClick); }); |
| --- |

## Event Targets, Bubbling and Propagation

Kuomet callback funkcija yra pašaukiama, šaukimo metu ji gauna argumenta event. Event yra objektas, kuris savyje turi daugybe informacijos, mes galime pamatyti, koks tai buvo iventas, kas jį išaukė, kokioje ekrano lokacijoje jis vyko, ir ant kokio elemento jis buvo iššauktas. Event.target mes gali nustatyti, ant kurio elemento eventas buvo pašauktas.

Yra dvi event savybės į kurias reikia atkreipti dėmesį, tai target ir currentTarget.

event.taget tai yra elementas ant kurio butent ir įvyko tas eventas, o currentTarget yra elementas prie kurio buvo pririštas tas eventas.

Pvz.

| window.addEventListener("click", function (event) {  console.log(event.target); *// button*  console.log(event.currentTarget); *// window*  }); |
| --- |

Kuomet paspaudžiame pelės mygtuka ant button elemento, šitas eventas trigerinasi, kad paspaudei ant mygtuko, paskiau ant tevinio elemento, tuomet body ir t.t. iki pat viršaus. Šitas procesas vadinasi event **bubbling**, kuomet eventas nusiridena per visus DOM lygmenius į viršų Bet tuo pačiu šita elgsena mums leidžia vienu metu užregistruoti kelis event listner’ius tarkim ant window (kas reiškia kad jis pasileis ant bet kur, visame lange) ir ant mygtuko.

Kai spaudžiame click ant pačio mygtuko, abudu eventai pasileidžia, tiek registruotas ant mygtuko, tiek and window, todėl, kad techniškai paspaudėte ant abiejų vienu metu. Šitas efektas kai vienu metu pasileidžia keli eventai vadinamas **propagation**.

Sustapdyti propagation mes galime e.stopPropagation() event metodu ant žemiausiai DOM medyje esančiu elemento, ir tuomet jis sustabdo event bubbling efekta. .

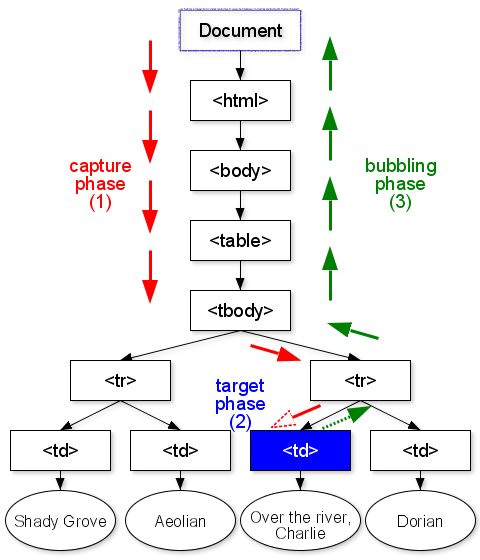
Pvz. Propagation: Paspaudus ant Click me! pasileis visi eventai.

| <div onClick={(e) => console.log('outer div')}>  <div onClick={(e) => console.log('middle div')}>  <div onClick={(e) => console.log('innermost div')}>  Click me!  </div>  </div> </div> |
| --- |

Pvz. Paspaudus ant Click me! Kadangi nurodėme stopPropagation pasispaus tik vienas.

| <div onClick={(e) => console.log('outer div')}>  <div onClick={(e) => console.log('middle div')}>  <div onClick={(e) => e.stopPropagation()}>  Click me!  </div>  </div> </div> |
| --- |

Vizualus pvz event bubbling.



Paspaudus ant elemento pirmiausia by default eventai pasileidžia nuo žemiausio ir bubblina į viršų į aukščiausia. Norėdami apkeisti tvarka, tam galime perduoti parametra. capture: true kuomet pridedame event listeneri.

| window.addEventListener("click", function (event) {  console.log("YOU CLICKED THE WINDOW");  console.log(event.target);  console.log(event.type);  console.log(event.bubbles); }, {capture: true }); button.addEventListener("click", function() {  console.log('You clicked on button') }); |
| --- |

Tuomet jei paspausite ant mygtuko, pirmas eventas bus window, o tik po to mygtuko.