wantsome
The friendly IT Academy

Saptamana 10

Partea 1

Programare Front-End

1. Events

Event (*Eveniment*)

- se petrece in urma producerii (completarii) unei actiuni

- **evenimentele** se pot petrece in urma terminarii unor anumitor faze din *ciclul de viata al unei pagini web* sau atunci cand utilizatorul interactioneaza cu elementele HTML din cadrul paginii web

  Exemple:

  - incarcarcarea unei pagini a fost finalizata
  - textul din cadrul unui input a fost schimbat de catre utilizator
  - utilizatorul a apasat un anumit button (***click***)

Event

- de cele mai multe ori, atunci cand un eveniment se petrece, ca developeri, vrem sa “executam” un anumit task functional, sa facem “ceva”; in acest sens, **HTML** ne pune la dispozitie **atribute** pentru specificarea ***event handlers*** care accepta ca valori functii sau cod JS

Sintaxa:

```
<element event='cod JS'>
```

Exemple:

```
<button
    onclick="document.getElementById('time').innerHTML = (new Date()).getHours() + ‘:’ + (new Date()).getMinutes()">
       Afiseaza ora
</button>
```

```
<button onclick="this.innerHTML = same code as upper ">Afiseaza ora</button>
```

Event

- in cele mai des intalnite situatii, *task*-urile functionale pe care vrem sa le executam atunci cand un eveniment s-a petrecut sunt destul de complexe, ceea ce inseamna ca avem nevoie de mai multe linii de cod JS

- specificarea tuturor acestor linii de cod in valoarea unui atribut care inregistreaza un *event handler,* ar produce un cod foarte greu de citit

- din acest motiv, pentru astfel de situatii, task-ul functional se incapsuleaza intr-o functie care este atasata ca si apel in valoarea unui astfel de atribut

Exemplu:

```
<button onclick="displayDate()">Afiseaza ora</button>
function displayDate() {
	const d = new Date();
	document.getElementById('time').innerHTML = d.getHours() + ‘:’ + d.getMinutes();
}
```

1.2 Event Types

Tipuri de evenimente

- exista o varietate destul de mare de evenimente asociate diferitor tipuri de elemente HTML
- printre cele mai des folosite, intalnim:

onchange - un element a suferit o schimbare

onclick - utilizatorul a dat click pe un element

onmouseover - utilizatorul a trecut cu mouse-ul peste un element

onmouseout - utilizatorul a inlaturat mouse-ul de pe un element

onkeydown - utilizatorul a apasat o tasta

onload - browser-ul a terminat de incarcat pagina HTML

...

Exemple:

https://www.w3schools.com/js/js_events_examples.asp

Tipuri de evenimente

- o intreaga lista a evenimentelor care pot fi captate poate fi gasita la urmatoarele referinte:

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

https://developer.mozilla.org/en-US/docs/Web/Events

1.3 Event Handlers, Event Listeners

Event Listeners, Event handlers

- atunci cand vine vorba de evenimente, avem doua concepte implicate:
  - **event listener** - atunci cand inregistram o functionalitate (functie) spre a fi executata in momentul in care se petrecere un eveniment specific, specificam/inregistram un ***listener*** cu ajutorul unor instructiuni
    - un listener este alcatuit din:
      - denumirea evenimentului la care engine-ul trebuie sa “asculte”, sa reactioneze
      - un **event handler**
  - **event handler** - functia care trebuie executata in momentul in care a avut loc un eveniment specificat in cadrul unui *listener*

Exemplu:

```
<button onclick="displayDate()">Afiseaza ora</button>
```

Modalitati de specificare a unui Event Listener

1. prin intermediul atributelor HTML

```
<button onclick="displayDate()">Afiseaza ora</button>
```

1. prin intermediul JS si specificarea *listener*-ului ca si metoda a elementului selectat

```
someElement.onclick = function() {
//codul corespunzator event handler-ului
};
```

1. prin intermediul JS si utilizarea metodei **`addEventListener()`**

```
someElement.addEventListener("mouseup", handleMouseUp, false);
```

Event flow

Event Flow

Event flow-ul se refera la ordinea in care sunt declansate evenimentele propagate de catre un element care se afla in interiorul altui element parinte

```
<ul>
    <li><a href="..."><img src="..." alt=""></a>
    <li><a href="..."><img src="..." alt=""></a>
    <li><a href="..."><img src="..." alt=""></a>
</ul>
```

Exemplu: Un click pe imagine nu genereaza evenimentul *click* doar pentru elementul HTML corespunzator imaginii ci si pentru elementul <a> care o incapsuleaza si pentru elementul <li>, progresiv, pana la *window object*. In acest exemplu imaginea poate fi numita ca fiind **event target,** reprezentand elementul de la care incepe propagarea.

Event Flow

In acest proces, propagarea este bi-directionala de la ***window*** catre ***event target*** si inapoi. Astfel, exista 3 faze:

1. De la **window catre event target** - **CAPTURE PHASE**
1. **Event target**-ul (cand ajunge la el) - **TARGET PHASE**
1. De la **event target catre window** - **BUBBLE PHASE**

Event Flow

Event CAPTURE PHASE

In aceasta faza doar capture listeners sunt chemati, adica doar cei care au cel de-al 3-lea parametru ***true***, precum in exemplul:

```
el.addEventListener('click', listener, true)
```

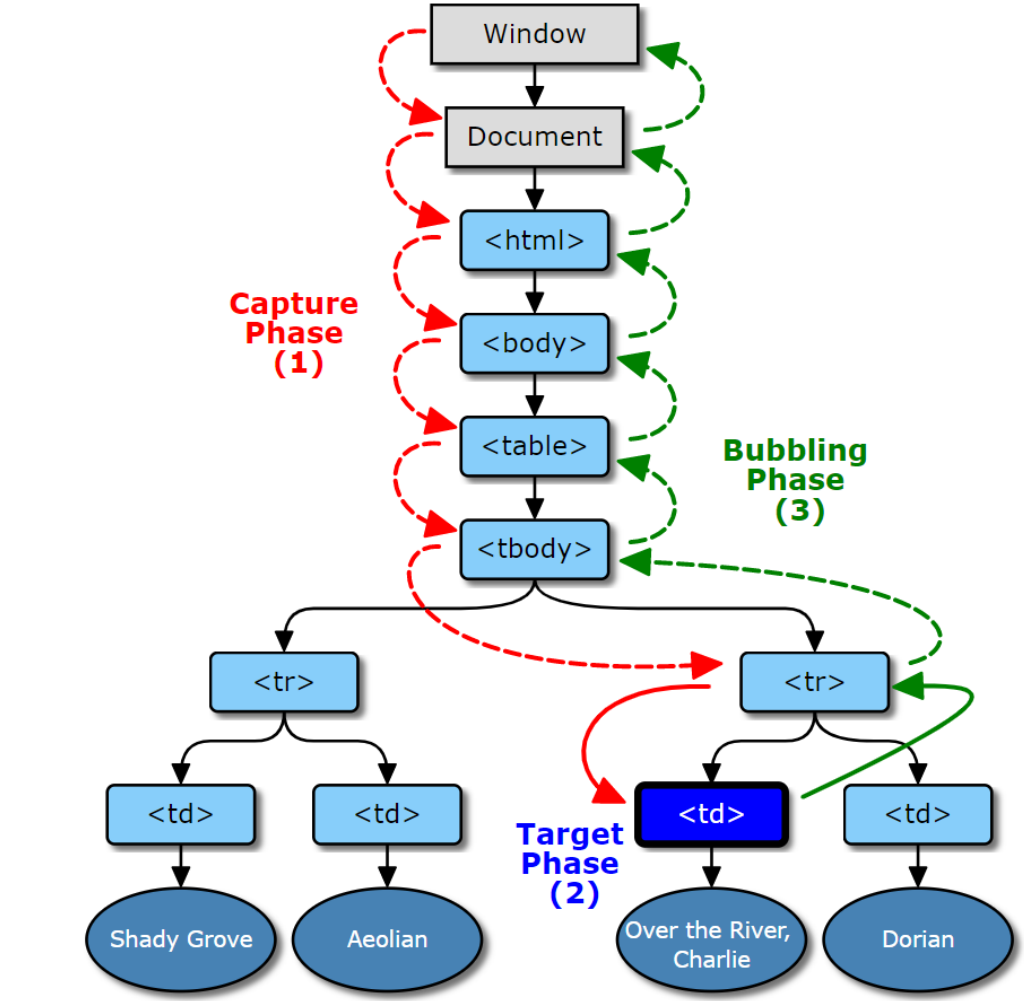
Daca nu specificam acest parametru, *by default* va fi **false**

Event TARGET PHASE

In aceasta faza **toti listener-ii inregistrati pe un anumit event target** vor fi invocati, indiferent de valoarea celui de-al 3-lea parametru

Event BUBBLING PHASE

In aceasta faza vor fi chemati toti listenerii pentru care valoarea celui de-al 3-lea parametru este **false**



Event Flow

https://codepen.io/oviduzz/pen/wLGbpj?editors=1111

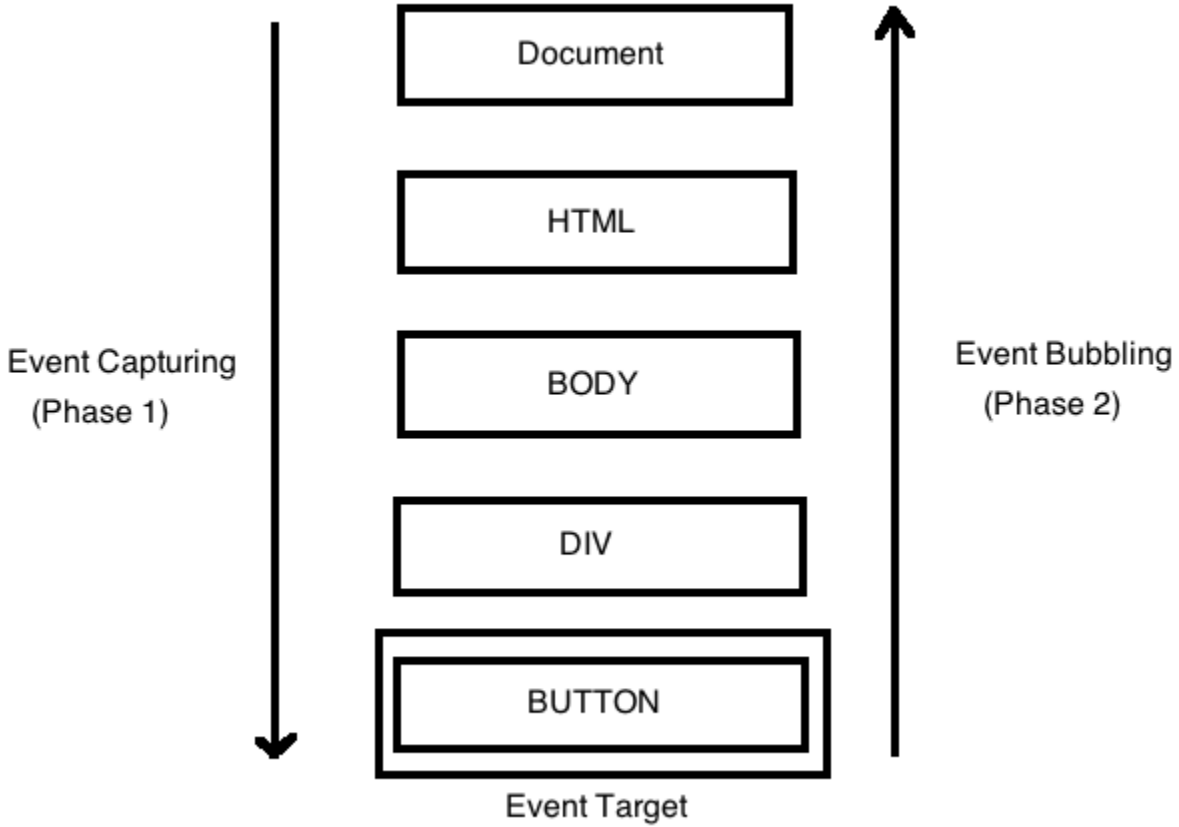
Button > div > body > html > Document - **Event Bubbling**

Stop event Bubbling:
Daca vrem sa oprim event bubbling-ul sau propagarea avem metoda de event.**stopPropagation**(). Aceasta metoda **opreste** tot *flow*-ul pana la ultimul parinte.

Event capturing

Daca adaugam al 3-lea parametru, cu valoarea true, vom observa cum prima data apare in consola valoarea trimisa de catre *listener*-ul inregistrat pe parinte, apoi cea trimisa de catre *listener*-ul inregistrat pe copil.

Acum ordinea va fi *document > html > body > div > button*



event bubbling cool expl

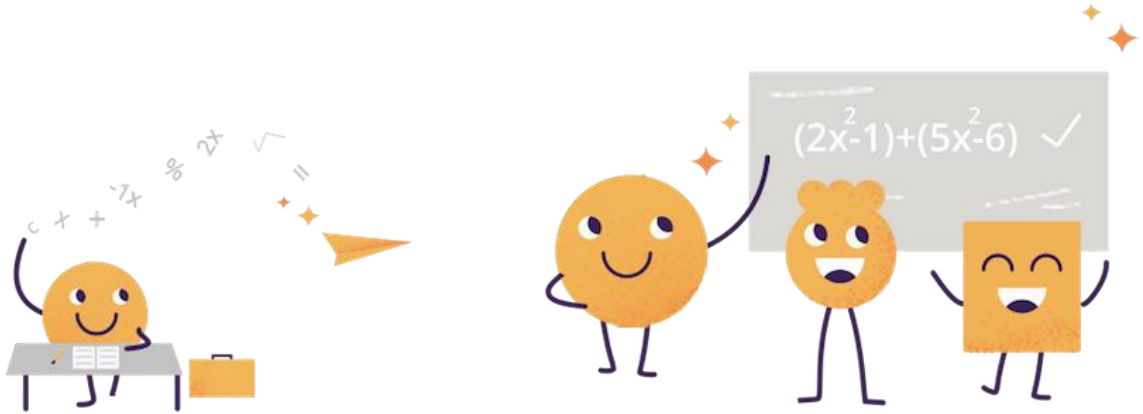
1.2 Custom Events

Definirea si captarea unor evenimente *custom*

https://developer.mozilla.org/en-US/docs/Web/Guide/Events/Creating_and_triggering_events

PRACTICE: **JavaScript Events**

http://bit.do/ex1Event
http://bit.do/ex2Event
http://bit.do/ex3Event



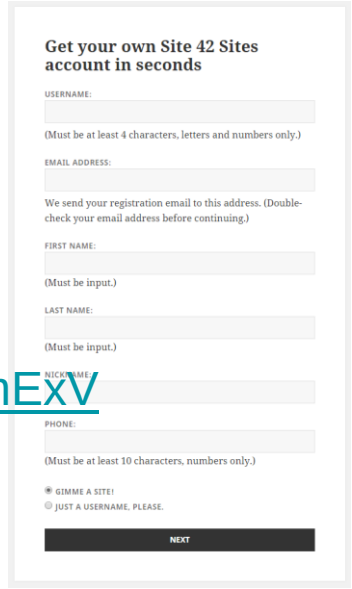
PRACTICE: **Forms Validation**

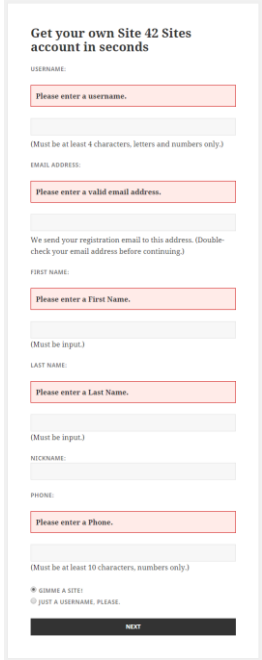
Cerinta:

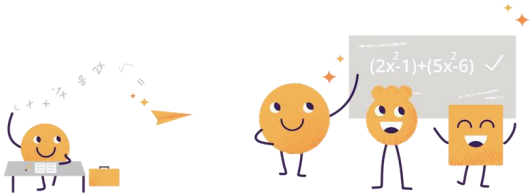
Folosind *DOM Manipulation* si *Events*, implementati *form*-ul si functionalitatea schitata in urmatoarele *mock*-uri:

- https://www.screencast.com/t/5nlaalKBxQ
- https://www.screencast.com/t/xVngJJl19J

http://bit.do/formExV







wantsome
The friendly IT Academy