JavaScript events in detail

Događaji u jeziku JavaScript

Ako postavimo "click" event listener na element, što se dešava ako kliknemo na *dijete* tog elementa?

```
<div class="show-details">
  <img src="https://s3-us-west-2.amazonaws.com/s.cdpn.io/1083533/forward-arrow.p
  <span>Show details</span>
</div>
```

```
const detailToggle = document.querySelector('.show-details');
detailToggle.addEventListener('click', toggleVisibility);
```

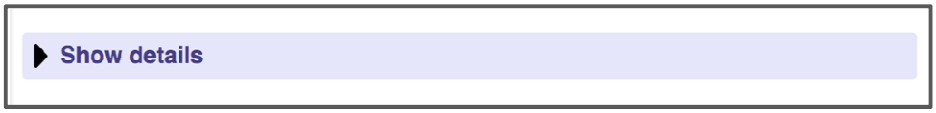
▶ Show details

Događaji u jeziku JavaScript

Primjer: Ako kliknemo na `<img>`, da li će se izvršiti funkcija `toggleVisibility`? (CodePen)

```
<div class="show-details">
  <img src="https://s3-us-west-2.amazonaws.com/s.cdpn.io/1083533/forward-arrow.p
  <span>Show details</span>
</div>
```

```
const detailToggle = document.querySelector('.show-details');
detailToggle.addEventListener('click', toggleVisibility);
```

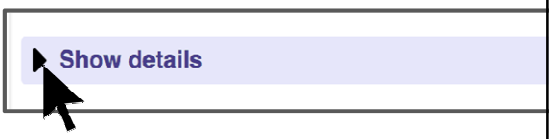


Događaji u jeziku JavaScript

DA, klik postavljen na element će sivršiti kao kliknete na dijete tog elementa

Ako postavimo click event listener na `div`, i kliknemom na `img` unutar `div`, izvršiće se event listener.

(CodePen)




```
<div class="show-details">
  <img src="https://s3-us-
  <span>Show details</span
</div>
```

Event.currentTarget vs target

```
function toggleVisibility(event) {
  const theElementClicked = event.target;
  const theElementTheEventIsTiedTo = event.currentTarget;
```

Možete pristupiti elementu na koji je kliknuto ili elementu za koji je zakačan event listener :

- ***event***.`target`: element nak loji je kliknuto / "dispatched the event" (moguće dijete target-a)
- ***event***.`currentTarget`: element za koji je originalni event handler zakačen.

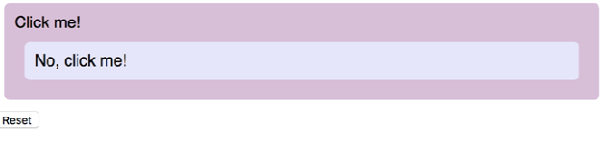
Više event listener-a

Što ako imamo event listener-e i za element i za dijete tog elementa?

- Da li se oba izvršavaju?
- Koji se prvi izvršava?

```
<div id="outer">
  Click me!
  <div id="inner">
    No, click me!
  </div>
</div>
```

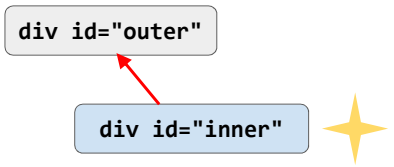
```
const outer = document.querySelector('#outer');
const inner = document.querySelector('#inner');
outer.addEventListener('click', onOuterClick);
inner.addEventListener('click', onInnerClick);
```



(CodePen)

Event bubbling

- Oba se okidaju ako kliknemo na unutrašnji element
- Podrazumijevano, event listener unutrašnje (inner-most) elementa prvi se izvršava

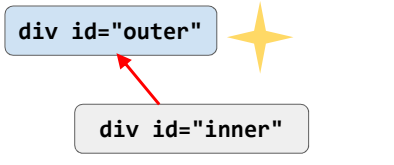



```
<div id="outer">
  Click me!
  <div id="inner">
    No, click me!
  </div>
</div>
```

Ovaj poredak događaja (inner-most to outer-most) je poznat kao **bubbling**. (CodePen)

Event bubbling

- Oba se okidaju ako kliknemo na unutrašnji element
- Podrazumijevano, event listener unutrašnje (inner-most) elementa prvi se izvršava




```
<div id="outer">
  Click me!
  <div id="inner">
    No, click me!
  </div>
</div>
```

Ovaj poredak događaja (inner-most to outer-most) je poznat kao **bubbling**. (CodePen)

stopPropagation()

Možete zaustaviti prenošenje događaja naviše (bubbling up) pozivom metoda ***event***`.stopPropagation()`:

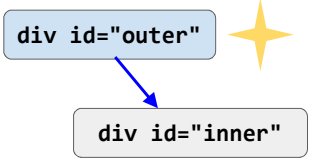
```
function onInnerClick(event) {
  inner.classList.add('selected');
  console.log('Inner clicked!');
  event.stopPropagation();
}
```

Deatlji : default behavior i stopPropagation

Event capturing

Ako želimo da prenošenje ide u suprotnom smjeru, dodajte treći parametar (3rd parameter) u `addEventListener`:

```
event.addEventListener(
    'click', onClick, { capture: true} );
```



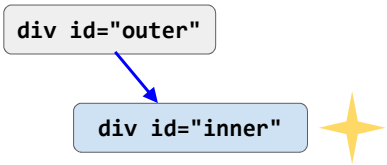

```
<div id="outer">
  Click me!
  <div id="inner">
    No, click me!
  </div>
</div>
```

Ovaj poredak događaja (outer-most to inner-most) je poznat kao **capturing**. (CodePen)

Event capturing

Ako želimo da prenošenje ide u suprotnom smjeru, dodajte treći parametar (3rd parameter) u `addEventListener`:

```
event.addEventListener(
    'click', onClick, { capture: true} );
```




```
<div id="outer">
  Click me!
  <div id="inner">
    No, click me!
  </div>
</div>
```

Ovaj poredak događaja (outer-most to inner-most) je poznat kao **capturing**. (CodePen)

`stopPropagation()`

Možemo korstiti ***event*.`stopPropagation()` i u capture-order:

```
function onOuterClick(event) {
  outer.classList.add('selected');
  console.log('Outer clicked!');
  event.stopPropagation();
}
```

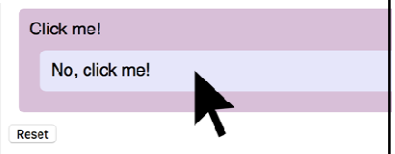
Detalji: default behavior i `stopPropagation`

Neki tehnički detalji...

Ispod haube...

Browser prolazi kroz obje faze (capture faza i bubbling faza) kada se realzuje događaj:

```
<html>
  <head>
    <meta charset="utf-8">
    <title>JS Events: Two event listeners</title>
  </head>
  <body>
    <div id="outer">
      Click me!
      <div id="inner">
        No, click me!
      </div>
    </div>
    <button>Reset</button>
  </body>
</html>
```

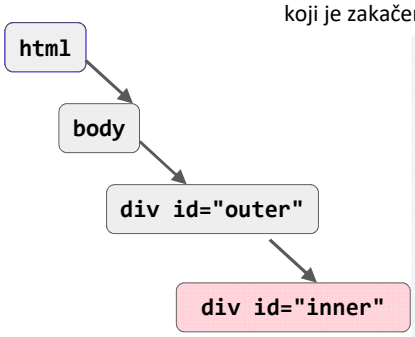


Ako kliknemo na div sa id="inner"...

Ispod haube...

Browser kreira tzv. "**propagation path**" za target tj. listu predala sve do korijena (w3c)

(`target` ovdje ima značenje "ono na što je kliknuto"; ne mora biti element za koji je zakačen event listener)

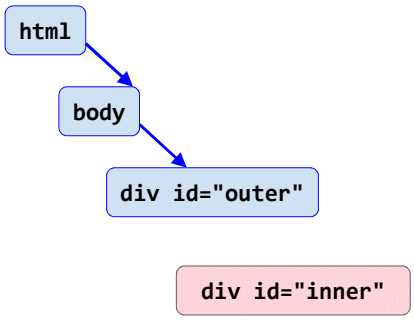



```
<html>
  <head>
    <meta charset="utf-8">
    <title>JS Events: Two event listeners</tit
  </head>
  <body>
    <div id="outer">
      Click me!
      <div id="inner">
        No, click me!
      </div>
    </div>
    <button>Reset</button>
  </body>
</html>
```

"Capture phase"

Browser počinje od vrha liste i poziva svaki event listener za koji je `capture="true"`, sve dok ne dođe do target-a.

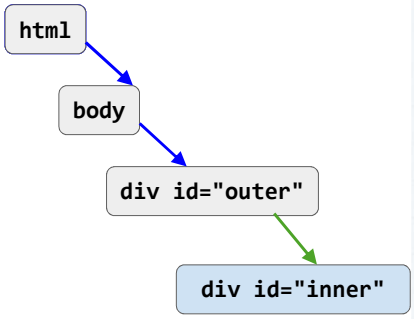
Ovo je tzv. "**capture phase**" (w3c)




```
<html>
  <head>
    <meta charset="utf-8">
    <title>JS Events: Two event listeners</t
  </head>
  <body>
    <div id="outer">
      Click me!
      <div id="inner">
        No, click me!
      </div>
    </div>
    <button>Reset</button>
  </body>
</html>
```

"Target phase"

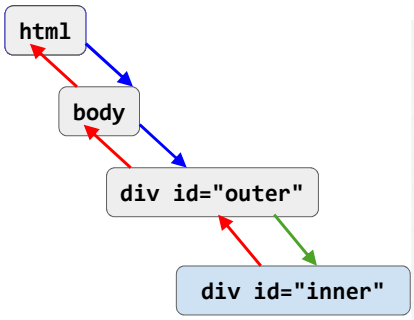
Zatim browser poziva sve event listener-e koji su postavljeni na samom target-u. To je "target phase" (w3c)




```
<html>
  <head>
    <meta charset="utf-8">
    <title>JS Events: Two event listeners</tit
  </head>
  <body>
    <div id="outer">
      Click me!
      <div id="inner">
        No, click me!
      </div>
    </div>
    <button>Reset</button>
  </body>
</html>
```

"Bubble phase"

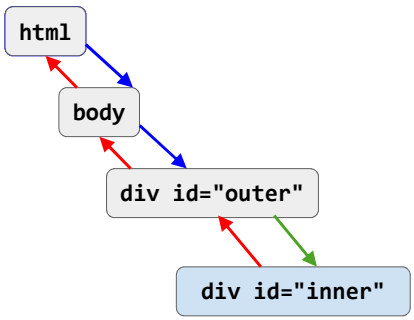
Ako je za event postavljeno `bubbles=true` (npr, pogledajte `click`) browser ide unazad kroz listu i poziva event listener koji nije okinut na capture. Ovo je "**bubble phase**" (w3c)




```
<html>
  <head>
    <meta charset="utf-8">
    <title>JS Events: Two event listeners</tit
  </head>
  <body>
    <div id="outer">
      Click me!
      <div id="inner">
        No, click me!
      </div>
    </div>
    <button>Reset</button>
  </body>
</html>
```

stopPropagation()

`stopPropagation()` u stvari zaustavlja ostatak ovog trofaznog procesa tj. sprečava da se ostatak koda izvrši



Praksa

Don't worry about:

- Vjerovatno nećete morati da koristite capture order – skoro uvijek koristite bubbling
- Nije baš obavezno da znate kako browser prolazi kroz "capture phase", "target phase" i "bubble phase"

Do worry about:

- **Morate razumijeti kao radi bubbling**
- `stopPropagation()` ponekad baš pomaže