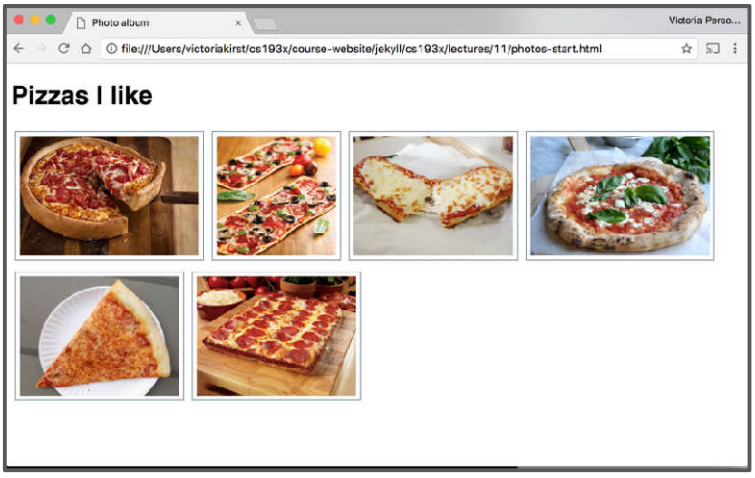
Agenda

Teme:

- Keyboard events
- Mobile events
- Jednostavne CSS animacije
- Klase i objekti
- `this` i `bind`

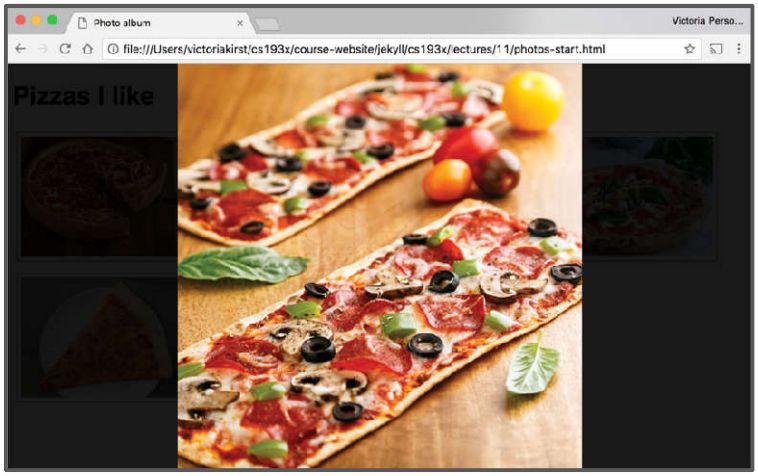
Primjer: Photo Album

Dodaćemo neke nove stvari ovom albumu :



Primjer: Photo Album

Dodaćemo neke nove stvari ovom albumu:



Pregledajmo fajlove:
photo-start.html
photo.js
photo.css

General setup

```
<body>
  <h1>Pizzas I like</h1>
  <section id="album-view">
  </section>

  <section id="modal-view" class="hidden">
  </section>
</body>
```

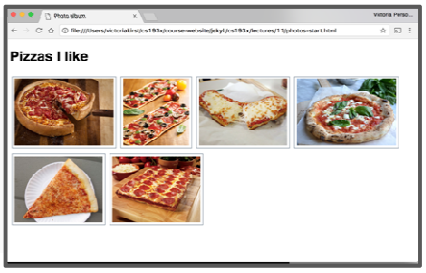
photo.html sadrži oba "ekrana:

- album view: thumbnails za svaku fotografije
- "modal" view: jedna fotografija iznad poluprozirne (semi-transparent) crne podloge – sakriven (by default)

CSS: Album

photo.css: CSS za album view je jednostavan:

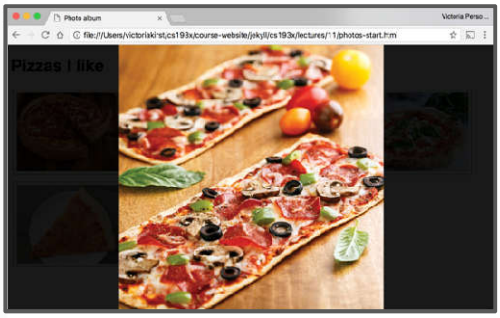
```
#album-view img {
  border: 1px solid slategray;
  margin: 5px;
  padding: 5px;
  height: 150px;
}
```



CSS: Modal

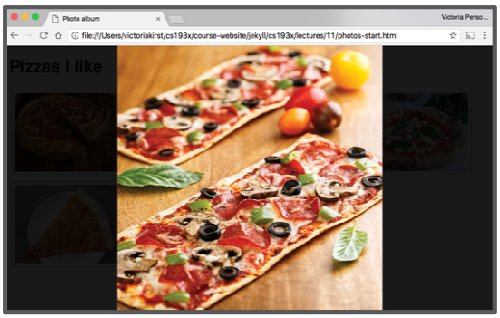
Modal view je malo složeniji:

```
#modal-view {
  position: absolute;
  top: 0;
  left: 0;
  height: 100vh;
  width: 100vw;

  background-color: rgba(0, 0, 0, 0.9);
  z-index: 2;

  display: flex;
  justify-content: center;
  align-items: center;
}
```



CSS: Modal image

```
#modal-view img {
  max-height: 100%;
  max-width: 100%;
}
```

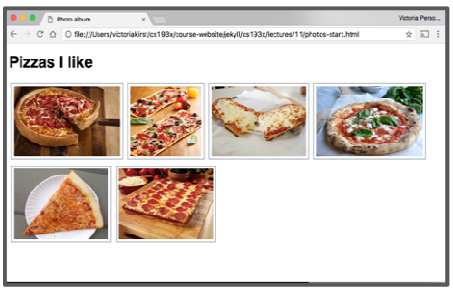


Veličina slike je ograničena na širinu i visinu roditelja (`#modal-view` čija je visina i širina podešena na dimenzije viewport)

CSS: Hidden modal

```
<body>
  <h1>Pizzas I like</h1>
  <section id="album-view">
  </section>

  <section id="modal-view" class="hidden">
  </section>
</body>
```

```
#modal-view.hidden {
  display: none;
}
```



Oba prikaza (album view i modal view) se nalaze u HTML-u, ali je model view postavljen `display: none;` pa se ne prikazuje.

Globalni niz fotografija

```
<head>
  <meta charset="utf-8">
  <title>Photo album</title>
  <link rel="stylesheet" href="css/photo.css">
  <script src="js/photo-list.js" defer></script>
  <script src="js/photo.js" defer></script>
</head>
```

```
const PHOTO_LIST = [
  'images/deepdish.jpg',
  'images/flatbread.jpg',
  'images/frenchbread.jpg',
  'images/neapolitan.jpg',
  'images/nypizza.jpg',
  'images/squarepan.jpg'
];
```

photo-list.js: globalni niz PHOTO_LIST koji sadrži stringove koji su imena fotografija (putanje do fotogtafija).

Photo thumbnails

```
function createImage(src) {
  const image = document.createElement('img');
  image.src = src;
  return image;
}
```

```
const albumView = document.querySelector('#album-view');
for (let i = 0; i < PHOTO_LIST.length; i++) {
  const photoSrc = PHOTO_LIST[i];
  const image = createImage(photoSrc);
  image.addEventListener('click', onThumbnailClick);
  albumView.appendChild(image);
}
```

photo.js: Popunjavanje početnog albuma: ciklus po nizu PHOTO_LIST i nadovezivanj e <img> na #album-view.

Klik na sliku

```
function onThumbnailClick(event) {
  const image = createImage(event.currentTarget.src);
  modalView.appendChild(image);
  modalView.classList.remove('hidden');
}
```

Kada korisnik klikne na sliku:

- Kreiarmo novi <img> tag sa istim atributom src
- Nadovežemo novi <img> na #modal-view
- Postavimo da je #modal-view vidljiv

Pozicioniranje modalnog pogleda

```
function onThumbnailClick(event) {
  const image = createImage(event.currentTarget.src);
  modalView.style.top = window.pageYOffset + 'px';
  modalView.appendChild(image);
  modalView.classList.remove('hidden');
}
```

Dodajemo liniju koda u JavaScript da i utvrdili poziciju na “top of the viewport” ne za “top of the screen”:

`modalView.style.top = window.pageYOffset + 'px';`

(Pogledajte `window.pageYOffset` mdn. Ili `window.scrollY`.)

Atribut `style`

Svakom `HTMLElement` može se doadti atribut `style` koji zadaje stil tog elemenata:

```
element.style.top = window.pageYOffset +
'px';
```

U opštem slučaju **ne treba koristiti ovo svojstvo**, jer je dodavanje i uklanjanje klasa pomoću `classList` bolji način za promjenu stila elementa

Ali ako mijenjamo CSS svojstvo na osnovu vrijednosti iz JavaScript-A, moramo postaviti atribut `style` neposredno .

No scroll na stranici

```
function onThumbnailClick(event) {
  const image = createImage(event.currentTarget.src);
  document.body.classList.add('no-scroll');
  modalView.style.top = window.pageYOffset + 'px';
  modalView.appendChild(image);
  modalView.classList.remove('hidden');
}
```

```
.no-scroll {
  overflow: hidden;
}
```

Zabranjujemo skrolovanje na stranici tako što postavimo `body { overflow: hidden; }` – ovo je jedan način da se to uradi.

Zatvaranje modalnog dijaloga

```
function onModalClick() {
  document.body.classList.remove('no-scroll');
  modalView.classList.add('hidden');
  modalView.innerHTML = '';
}
```

```
const modalView = document.querySelector('#modal-view');
modalView.addEventListener('click', onModalClick);
```

Kad korisnik klikne na modal view:

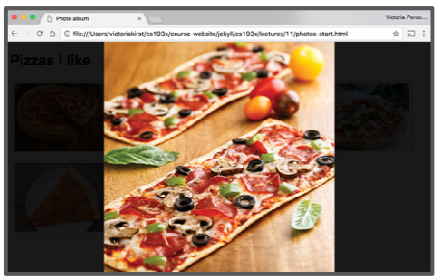
- Ponovo sakrivamo modal view
- Dopuštamo skrolovanje na stranici
- Uklanjamo veliku sliku pomoću `innerHTML = '';`

Navigacija tasterima

Kretanje kroz album

Dodajmo neke tzv. “keyboard events” za naš Modal View:

- Left arrow: prikazuje "i - 1“-vu fotografiju
- Right arrow: prikazuje "i +1“-vu fotografiju
- Escape key: zatvara dijalog



Kako osluškujemo “keyboard events”?

Keyboard events

Naziv događaja	Opis
`keydown`	Okida kada se pritisne taster (key is pressed). Ako držite pritisnut taster opet okida (mdn)
`keypress`	Okida kada se pritisne bilo koji **character** taster (slovo ili cifra). Nastavlja okidanje ako se drži pritsinut taster. (mdn)
`keyup`	Okida kada prestanemo da pritiskamo taster (mdn)

Možemo osluškivati keyboard events tako što dodamo event listener za `document`:

```
document.addEventListener('keyup', onKeyUp);
```

KeyboardEvent.key

```
function onKeyUp(event) {
  console.log('onKeyUp:' + event.key);
}
document.addEventListener('keyup',
onKeyUp);
```

Funkcije koje osluškuju događaje imajuparametar tipa `KeyboardEvent`.
Objekat tipa `KeyboardEvent` ima svojstvo `key`, koje čuva vrijendost ključa ako što je `"Escape"`

- List of key values

Vrijednosti za naš zadatak

Key string value	Opis
`"Escape"`	The Escape key
`"ArrowRight"`	The right arrow key
`"ArrowLeft"`	The left arrow key

Primjer: key-events.html

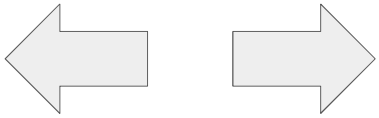
Rješenje:
photo-desktop-finished.html

Mobile?

Keyboard events su dobri za desktop, ali ne rade baš dobro na mobilnim uređajima.

Obično se za navigaciju koriste **gestovi**:

- **Left swipe** pokazuje sljedeću fotografiju
- **Right swipe** pokazuje prethodnu fotografiju



Next Previous

Mobile?

Keyboard events su dobri za desktop, ali ne rade baš dobro na mobilnim uređajima.

Obično se za navigaciju koriste **gestovi**:

- **Left swipe** pokazuje sljedeću fotografiju
- **Right swipe** pokazuje prethodnu fotografiju

Kako da implementiramo ove gestove?

Custom swipe events

- Ne postoje gesture events i jeziku JavaScript (yet).
- Dakle, ne postoji "Left Swipe" ili "Right Swipe" događajkoji bi mogli osluškivati. (Obratite pažnju da `drag` ne radi ono što želimo u ovom primjeru nitu radi na mobilnom)

Da bi dobili željeno ponašanje, moramo ga sami implementirati. Upoznajemo se sa još nekim događaijima:

- MouseEvent
- TouchEvent
- PointerEvent

MouseEvent

Događaj	Opis
`click`	Okida kada kliknete (click and release) (mdn)
`mousedown`	Okida kada pritisnete (click down) (mdn)
`mouseup`	Okida kada prekinete pristisak (release) (mdn)
`mousemove`	Okida neprestano dok se miše pomjera (mdn)

*__mousemove__ radi samo za desktop jer ne postoji koncept miša na mobilnom.

TouchEvent

Događaj	Opis
`touchstart`	Okida kada dodirenete ekran (mdn)
`touchend`	Okida kada podignete prst sa ekrana (mdn)
`touchmove`	Okida neprestano dok pomjeramo prst po ekranu (mdn)
`touchcancel`	Okida kada je tačka dodira "disrupted" (npr. ako browser nije potpuno siguran što se dešava) (mdn)

*__touchmove__ radi samo na mobilnom (Primjer)

clientX i clientY

```
function onClick(event) {
  console.log('x' + event.clientX);
  console.log('y' + event.clientY);
}
element.addEventListener('click', onClick);
```

MouseEvents imaju `clientX` i `clientY` :

- `clientX`: Relativna pozicija po x-osi u odnosu na lijevu ivicu browser-ovog viewport-a
- `clientY`: Relativna pozicija po y-osi u odnosu na gornju ivicu browser-ovog viewport-a

Implementiranje operacije drag

Kada korisnik pritisne taster miša ili dodirne element …

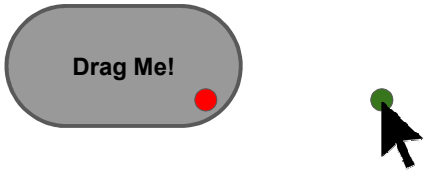
Implementiranje operacije drag

```
originX = 100;
```

Zapamtimo početnu poziciju.

Implementiranje operacije drag

```
originX = 100;
newX = 150;
```



Na `mousemove` / `touchmove`, pamtimo novu poziciju

Implementiranje operacije drag

```
originX = 100;
newX = 150;
```

Pomjeramo element za razliku između stare i nove pozicije.

Implementiranje operacije drag

I na operaciju podizanja (release)...

Implementiranje operacije drag

… prestajemo da osluškujemo `mousemove` / `touchmove`.

Dragging na mobilnom i na desktopu

Zara ne bi bilo lijepo da ne moramo da pišemo odvojene događaje za mobilne i za desktop?

PointerEvent

`PointerEvent`: "pointer" događaji rade isto i za mouse i za touch

- Nemojte ga pomiještai sa CSS svojstvom `pointer-events`
- **Note: U ovom slučaju umjesto smušenog objašnjenja na mdn** o `PointerEvent` pogledajte
  - A Google blog post on `PointerEvent`

`PointerEvent` je izveden iz `MouseEvent`, pa otuda ima `clientX` i `clientY`

`PointerEvent`

Događaj	Opis
`pointerdown`	Okida kada "pointer becomes active" (ddir ekerana ili pritisnut taster miša) (mdn)
`pointerup`	Okida kada pointer više nije aktivan (mdn)
`pointermove`	Okida neprestano dok se pointer pomjera (mouse move ili touch drag) (mdn)
`pointercancel`	Okida kada je "interrupted" (mdn)

*__`pointermove`__ radi i na mobilnom i na desktopu!
… osim…

Malo kontraverze!

`PointerEvent` **nije** implementiran na svim browserima:

- Provjerite na mdn

Polyfill biblioteka

Biblioteka polyfill sadrži kod koji imlementira podršku za browsere koji ne podržavaju Web API (natively implement a web API).

Postoji polyfill biblioteka za `PointerEvent`:
https://github.com/jquery/PEP

PointerEvent Polyfill

Da bi koristili PEP polyfill library, dodajmo sljedeći `script` tag u naš HTML:

```
<script src="https://code.jquery.com/pep/0.4.1/pep.js"></script>
```

I moramo da dodamo `touch-action="none"` u one oblasti za koje želimo da `PointerEvents` bude prepoznat*:

```
<section id="photo-view" class="hidden" touch-action="none">
</section>
```

*Na ovaj način kažemo browseru da ne želimo podrazumijevano ponašanje za djecu ovog elementa (npr.za mobilni ne želimo da prepozna uobičajeni "pinch to zoom" tip događaja) jer ih presrećemo pomoću PointerEvent. Ovo je obično CSS svojstvo, ali zbog ograničenja biblioteke polyfill moramo ga postaviti kao HTMKLatribut.

Pomjeranje elementa

Korostićemo CSS svojstvo `transform` da bi pomjerili element koji vučemo sa početne pozicije:

```
originX = 100;
newX = 150;
delta = newX - originX;
```


```
element.style.transform = 'translateX(' + delta + 'px)';
```

transform

`transform` je veoma moćno CSS svosjtvo koje nam dozvoljava translaciju, rotaciju, skaliranje i rastezanje elementa.

`transform: translate(x, y)`	Moves element relative to its natural position by ***x*** and ***y***
`transform: translateX(x)`	Moves element relative to its natural position horizontally by ***x***
`transform: translateY(y)`	Moves element relative to its natural position vertically by ***y***
`transform: rotate(deg)`	Rotates the element clockwise by ***deg***
`transform: rotate(10deg) translate(5px, 10px);`	Rotates an element 10 degrees clockwise, moves it 5px down, 10px right

Primjeri

translate vs position

Zar ne možemo koristiti pozicioniranje sa `relative` ili `absolute` da ostvarimo isti efekat kao i sa `translate`? Ima li razlike?

- `translate` je mnogo brži
- `translate` je optimizovan za animacije

Poređenje (članak):
- Absolute positioning (click "10 more macbooks")
- transform: translate (click "10 more macbooks")

Konačno, malo kodiranja!

preventDefault()

Za desktop postoji podrazumijevano ponašanje za povlačenje slike koje moramo onemogućiti pozivom event.preventDefault():

```
function startDrag(event) {
  event.preventDefault();
```

setPointerCapture()

Da bi osluškivali događaje pointera kada pointer izađe van ekrana, pozivamo `setPointerCapture` na meti za koju želimo da nastavimo praćenje:

```
event.target.setPointerCapture(event.pointerId);
```

Atribut `style` – još jednom

Atribut `style` ima **veći prioritet** od bilo kod CSS svojstva.

Da poništimo vrijednost postavljenu pomoću atributa `style`, postavljamo ga prazan string :

```
element.style.transform = '';
```

Sada se element može stilizovati u skladu sa pravilima iz CSS fajlova.

(requestAnimationFrame)

(Preskočli smo jedan važan dio koda koji nam dopušta da se pomjeranje obavlja glatko: `requestAnimationFrame`

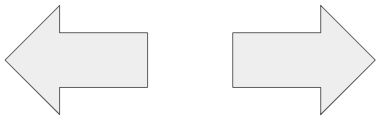
Međutim, upotreba `requestAnimationFrame` zahtijeva da razumijemo JavaScript event loop i funkcionalno programiranje.

CSS animacije

Uljepšavanje albuma

Dodajmo još animacija našem albumu.

- Ako prelazimo na prethodnu sliku, koristićemo **slide in from the left**
- Ako prelazimo na sljedeću sliku, koristićemo **slide in from the right**



Next Previous

Sintaksa za CSS animacije

```
@keyframes animation-name {
  from {
    CSS styles
  }
  to {
    CSS styles
  }
}
```

Primjeri

I zatim se postavi vrijednost za sljedeće CSS svojstvo :

`animation: animation-name duration;`

Primjer: Fade in

```
#album-view img {
  animation: fadein 0.5s;
}

@keyframes fadein {
  from {
    opacity: 0;
  }
  to {
    opacity: 1;
  }
}
```

Rezultat:
photo-mobile-finished.html