

Animácie

21.04.2022

Marek Nagy

Animácia

- pohyby, zmeny obrázku, ...
 - fps (frames per second)
 - cca 16 – 24 fps
- cez **CSS**
 - **transition**, **animation**
- cez **JS**
 - **setInterval()**, **requestAnimationFrame()**
 - **animate()**

CSS Transition

- prechody pomocou CSS
- pri zmene hodnoty CSS vlastnosti elementu
 - plynulý prechod zo starej hodnoty na novú
 - animovanie zobrazenia elementu v čase
 - t.j. nebude sa meniť skokom
- animáciu realizuje prehliadač

Výber vlastností a trvania

transition-property

- vlastnosti, ktorých zmena je animovaná prechodom

Budú sa realizovať 3 súbežné animácie.

```
div {  
  transition-property: width,height,top;  
  //transition-property: all;  
}
```

Jednoducho sa určia všetky CSS vlastnosti.

transition-duration

- čas realizácie prechodu
- v sekundách (**s**), alebo v (**ms**)

```
div {  
  transition-duration: 500ms;  
}
```

Celkový čas bez ohľadu, ako veľmi je CSS vlastnosť zmenená. Napr. zmena šírky.

Vývoj zmeny a odklad

transition-timing-function

- ako sa animuje prechod:

steps(n) – v n krokoch

linear – plynulý lineárny prechod

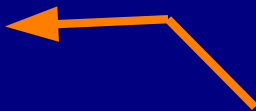
ease-in – pomalší nábeh

ease-out – pomalší dobeh

...

transition-delay

- odloženie začiatku animácie (s / ms)



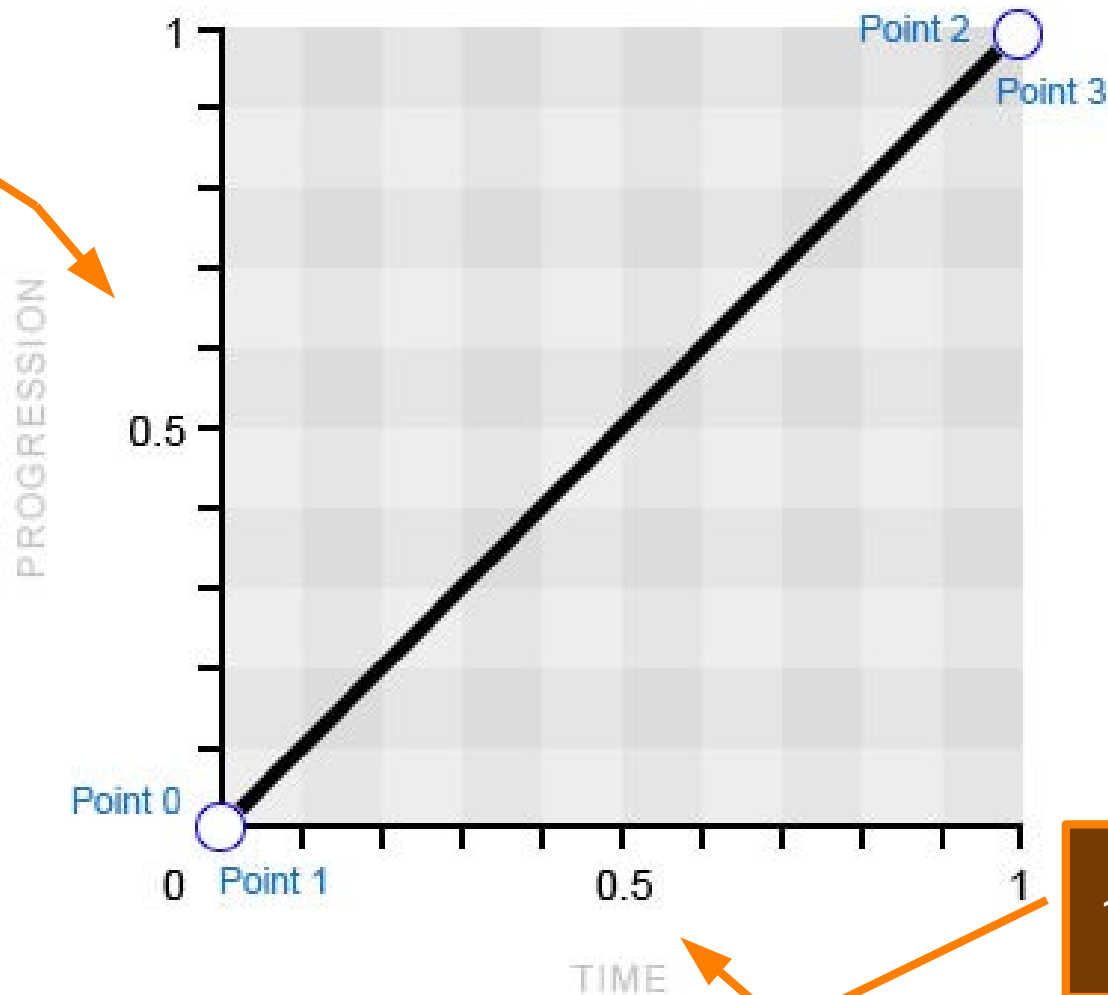
Animáciu realizuje prehliadač podľa svojich možností. Vývoj prechodu z hodnoty do hodnoty nemusí byť lineárny.

linear
cubic-bezier(0, 0, 1, 1)

P1 P2

Vo všeobecnosti stačí iba funkcia „cubic-bezier“. Dostane súradnice kontrolných bodov „Point 1“ a „Point 2“. Body 0 a 3 sú jednoznačne napevno určené.

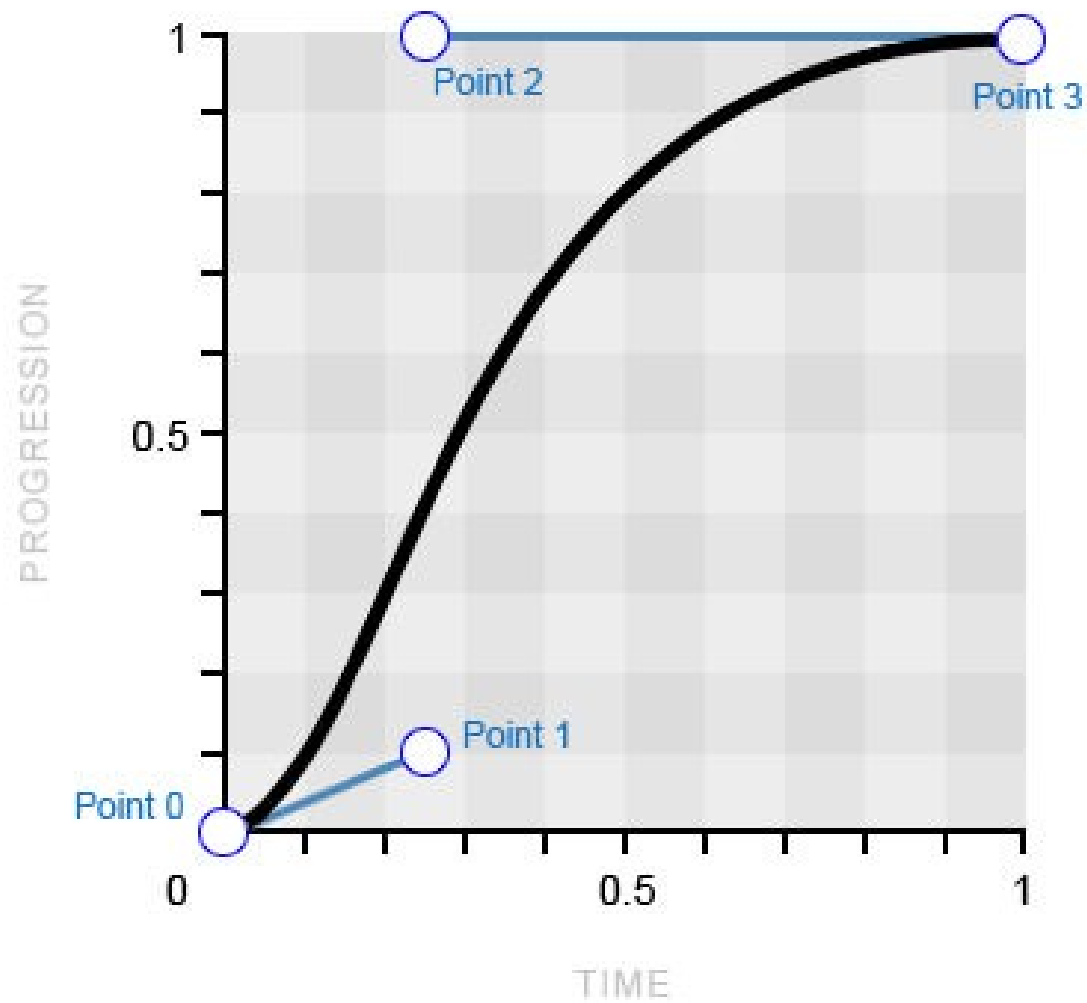
Normalizovaná hodnota. T.j. 1 zodpovedá cieľovej hodnote vlastnosti.



Normalizovaný čas. T.j. 1 zodpovedá cieľovému času prechodu.

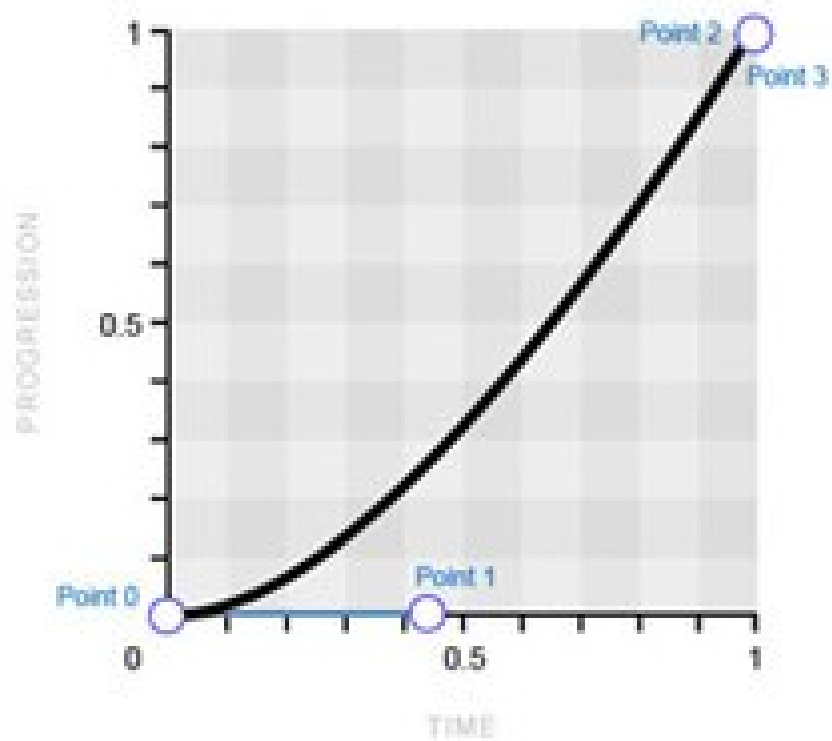
ease

cubic-bezier(.25, .1, .25, 1)



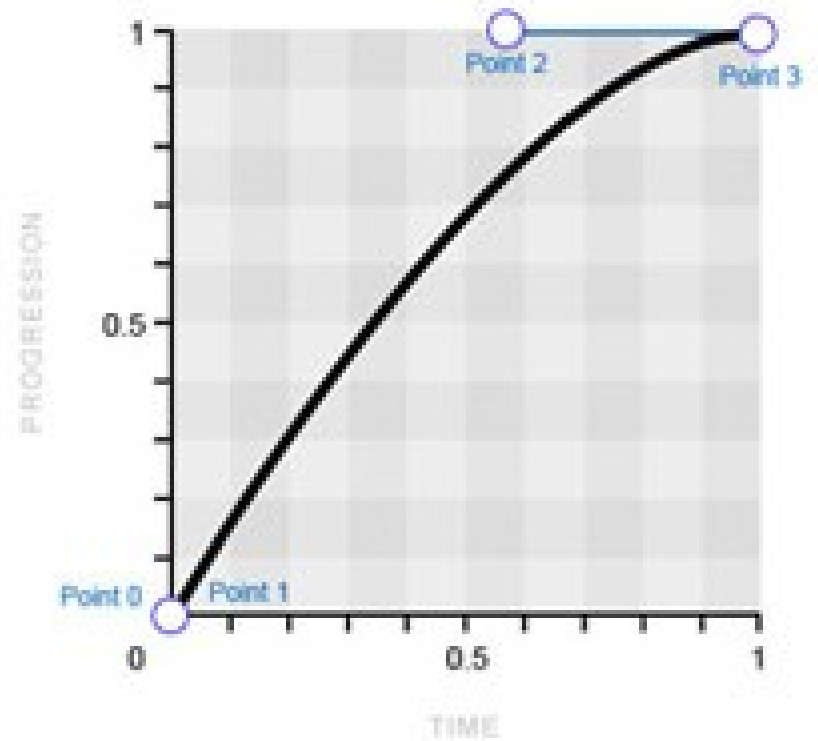
ease-in

cubic-bezier(.42, 0, 1, 1)



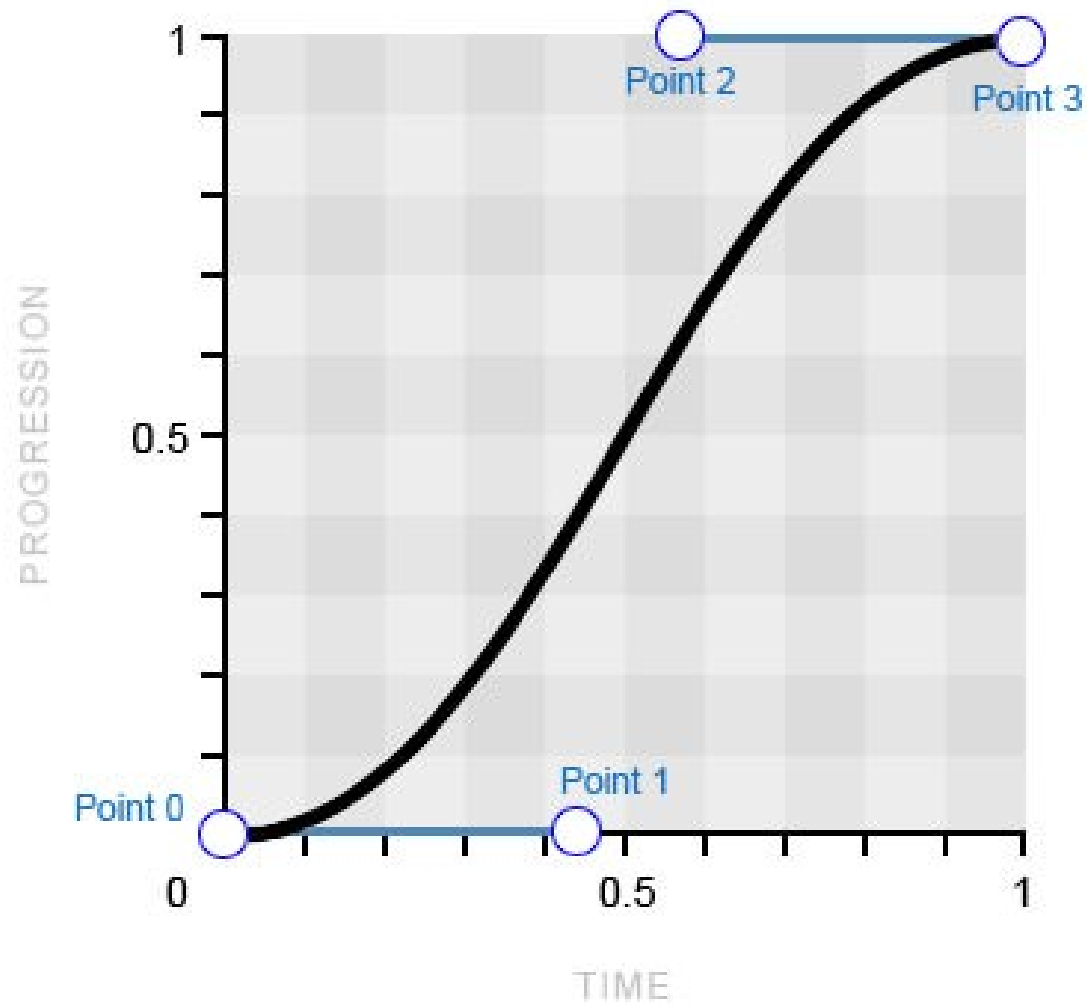
ease-out

cubic-bezier(0, 0, .58, 1)



ease-in-out

cubic-bezier(.42, 0, .58, 1)



Kompaktné nastavenie

transition

- všetky nastavenia prechodu naraz

```
div {  
    width: 100px;  
    transition: width 2s step(10) 100ms;  
}  
  
div:hover {  
    width: 300px;  
}
```

Udalosti

- generujú sa na danom DOM elemente
 - každá animovaná CSS vlastnosť generuje udalosti (individuálne animácie)
 - event.**propertyName** zodpovedá názvu animovanej CSS vlastnosti

transitionrun

- zaradené na vykonanie, ešte bude čakať (delay)

transitionstart

- po úvodnom čakaní, keď začne animovanie

transitionend

- keď skončí animácia
- možno reťaziť viaceré prechody

transitioncancel

- zrušená animácia
 - napr. zmena hodnôt vybraných css vlastností pre transition alebo display:none

CSS Animation

- animácie pomocou CSS
- riadená zmena CSS vlastností elementu
 - animácia na viac krokov
 - definovanie kľúčových momentov
 - animujú sa prechody medzi kľúčovými momentmi
- dá sa realizovať iba pre číselné CSS vlastnosti

@keyframes

- vytvorí fázy animácie
 - definuje kľúčové hodnoty-zmeny CSS vlastností
 - označené menom

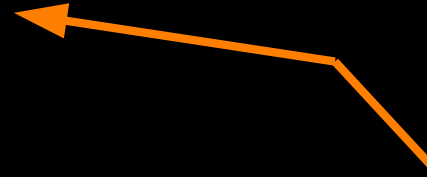
from → to

- zmena na novú hodnotu

0%, 1%, 50%, 100%

- postupná zmena na viaceré hodnoty


```
@keyframes example1 {  
  from {background-color: red;}  
  to {background-color: yellow;}  
}
```



„from“ vlastne
zodpovedá 0% a „to“
zodpovedá 100%.

```
@keyframes example2 {  
  0% {background-color: red;}  
  25% {background-color: yellow;}  
  50% {background-color: blue;}  
  100% {background-color: green;}  
}
```

Trvanie a vývoj zmeny

animation-name

- meno daného @keyframes

animation-duration

- trvanie animácie (s / ms)

animation-timing-function

- prechody (linear, steps(), ...)
- interaktívna ukážka:

<https://matthewlein.com/tools/ceaser>



Podobne ako
pri CSS Transition

Odklad a opakovanie

animation-delay

- oneskorenie

animation-iteration-count

- počet opakovaní (číslo / **infinite**)

animation-direction

- sled animácie pri opakovaní
- **normal**, **reverse**
- **alternate** (normal+reverse)
- **alternate-reverse** (reverse+normal)

Pozdržanie a CSS hodnoty

animation-play-state

- nastaví stav behu animácie (t.j. možnosť pozdržať)
- **paused, running**

animation-fill-mode

- animácia „uzamkne“ vybrané CSS vlastnosti do vlastnej réžie
 - t.j. animácia „nedovolí“ zmeniť dané CSS vlastnosti
- aké hodnoty bude prehliadač brať do úvahy pre CSS vlastnosti elementu **mimo behu animácie**?

none – vráti sa ku pôvodným hodnotám

forwards – zostanú „uzamknuté“ podľa ukončenej animácie

backwards – počas odkladu bude už aplikovaná prvá fáza

both – forwards+backwards

Kompaktné nastavenie

animation

- všetko dokopy
- max 8 hodnôt

```
div {  
  animation: example 5s linear 2s infinite alternate;  
}
```

Udalosti

- generujú sa na danom DOM elemente
 - event.**animationName** zodpovedá názvu animácie

animationstart

- keď začne animácia

animationend

- keď skončí animácia
- možnosť reťazenia animácií

animationiteration

- iba pri celých iteráciách

CSS animácia cez JS

- metódy DOM elementu:

animate()

- vytvorí objekt triedy Animation
- vytvorenie keyframe-ov (počas behu)
- štandardne je zadaný prvý a posledný keyframe

```
elem.animate ([  
  {width: '0px', height: '0px'},  
  {width: '100px', height: '100px'}  
], {  
  duration: 1000,  
  easing: 'linear',  
  fill: 'forwards'  
});
```

Pole keyframe-ov.

Štandardne stačí prvý
a posledný keyframe.
animácie.

getAnimations ()

- vráti pole aktívnych animácií ku elementu

animate()

- možné zadať aj viac keyframe-ov

```
let anim = elem.animate ([
  {offset: 0.0, width: '0px', height: '0px'},
  {offset: 0.7, width: '100px', height: '100px', easing: 'ease'},
  {offset: 1.0, width: '500px', height: '500px'}
], {
  duration: 1000,
  easing: 'linear',
  fill: 'forwards'
});
```

Určenie
lokálneho
prechodu
ku fáze.

Určuje pozíciu
keyframe-u
na intervale 0↔1.

Zaujímavé je ešte **composite**, ktoré umožní
napr. pričítavať údaje ku CSS vlastnostiam.
Čím možno spraviť animácie relatívne.

CSS vlastnosť a jej
postupnosť kľúčových
hodnôt.

Nastaví
sa iba čas
trvania.


- iné varianty volania

```
elem.animate ( [ {width: '0px'}, {width: '100px'} ], 1000 );
elem.animate ( {width: ['0px', '100px']}, 1000 );
```


Trieda Animation

playState

- indikátor stavu: **idle**, **running**, **paused**, **finished**



Pripojená animácia drží svoje CSS vlastnosti „zamknuté“. Ich pôvodná hodnota sa neberie do úvahy. Iba metódou `cancel()` sa odomknú (stav **idle**).

play()

- odštartuje animáciu od začiatku (defaultne sa spustí po vytvorení objektu)
- alebo pokračuje v prerušenej animácii

pause ()

- pozastavenie behu animácie

finish()

- „pretočí“ animáciu na koniec

cancel()

- zruší animovanie, stále je možné zadať `play()`, vhodné pri viacerých animáciách ku jednému elementu
- prejde do stavu **idle**

Trieda Animation

commitStyles ()

- aktuálne CSS hodnoty animácie sa nakopírujú priamo do CSS elementu
- lepšie sa nadväzujú animácie
- hlavne v súčinnosti s `fill: 'forwards'`

Ľahké reťazenie animácií. Pridané do prehliadačov len nedávno.

finished

- promise, ktorý signalizuje ukončenie animácie

oncancel, onfinish

- vlastné metódy (ku udalostiam), ktoré sa zavolajú pri zrušení a pri skončení

```
anim.onfinish = () => { // Možno spustiť zreťazenú animáciu ...  
}  
  
await anim.finished; // Počká na dokončenie
```

Trieda Animation

startTime

- čas odštartovania [ms]
- vzhľadom k spusteniu prehliadača
- možno nastaviť aj čas, kedy sa naplánuje štart animácie

currentTime

- čas bežiacej animácie [ms]
- možno zmeniť a posunúť tak vývoj animácie

playbackRate

- rýchlosť animácie
 - 1 zodpovedá normálnej rýchlosti
 - 1 reverzná animácia

Ďakujem za pozornosť