Napredni razvoj programske potpore za web

- predavanja -

2021./2022.

8. Jednostranične web-aplikacije

Vue.js

Creative Commons

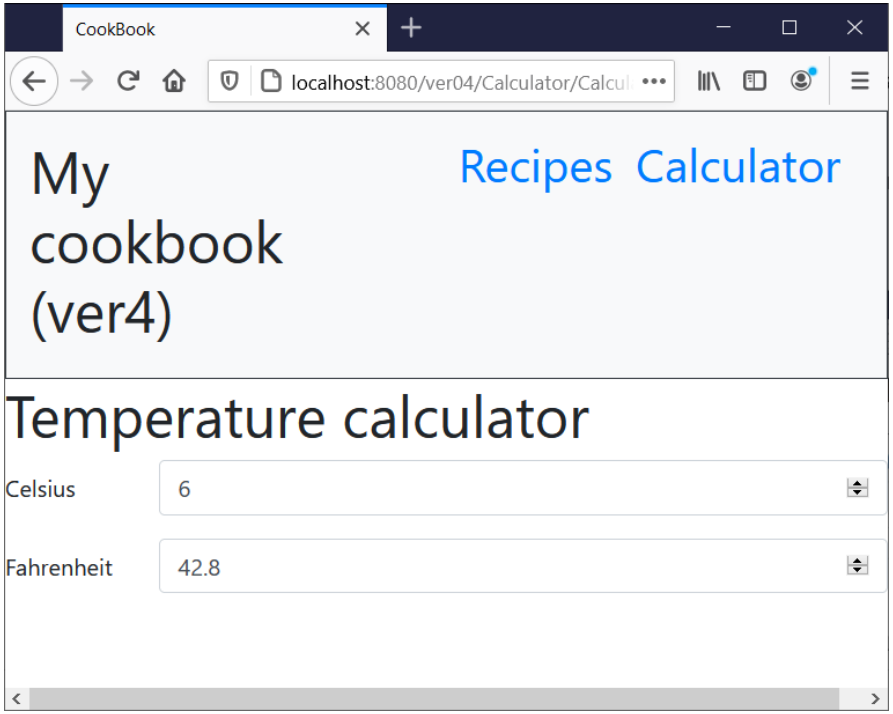
- slobodno smijete:
  - **dijeliti** — umnožavati, distribuirati i javnosti priopćavati djelo
  - **prerađivati** djelo
- pod sljedećim uvjetima:
  - **imenovanje:** morate priznati i označiti autorstvo djela na način kako je specificirao autor ili davatelj licence (ali ne način koji bi sugerirao da Vi ili Vaše korištenje njegova djela imate njegovu izravnu podršku).
  - **nekomercijalno:** ovo djelo ne smijete koristiti u komercijalne svrhe.
  - **dijeli pod istim uvjetima:** ako ovo djelo izmijenite, preoblikujete ili stvarate koristeći ga, preradu možete distribuirati samo pod licencom koja je ista ili slična ovoj.

U slučaju daljnjeg korištenja ili distribuiranja morate drugima jasno dati do znanja licencne uvjete ovog djela.
Od svakog od gornjih uvjeta moguće je odstupiti, ako dobijete dopuštenje nositelja autorskog prava.
Ništa u ovoj licenci ne narušava ili ograničava autorova moralna prava. *Tekst licence preuzet je s http://creativecommons.org/*

µSPA – ver4

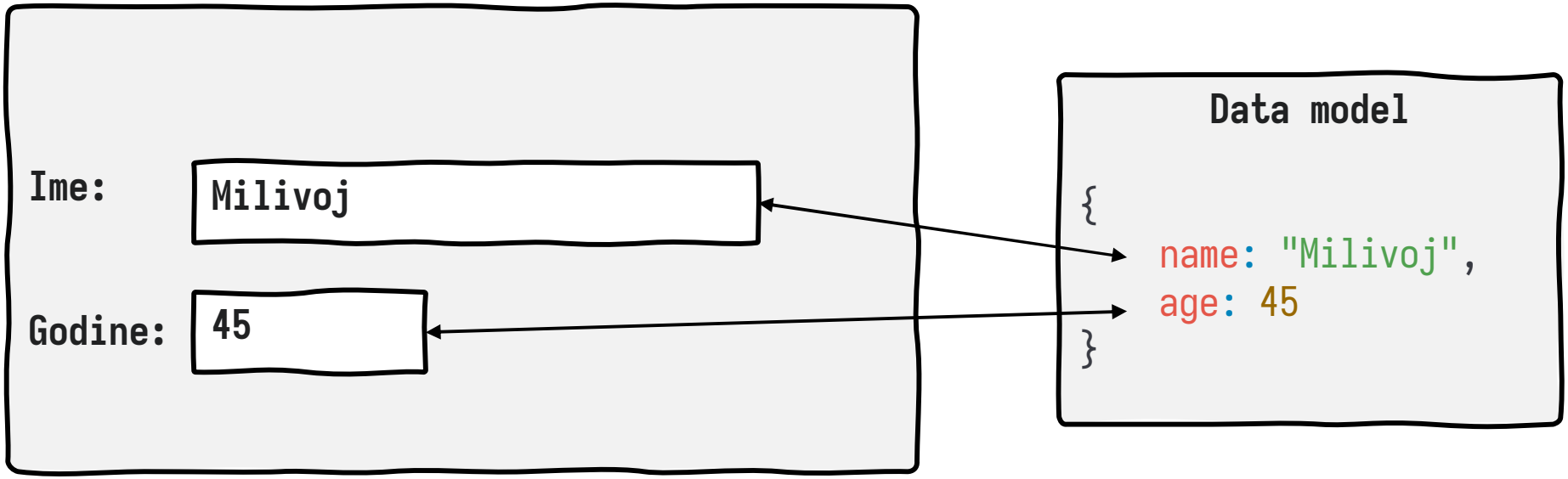
ver4

- Znamo:
  - Dinamički promijeniti sadržaj stranice
  - Presresti (i promijeniti) klik, ostvariti lokalno usmjeravanje i mijenjanje „stranica”
    - Koristimo vlastite oznake unutar HTML-a, npr. `data-link`
  - Razložiti kôd na module koji odgovaraju stranicama, stranice imaju jednostavne zadatke
  - Dohvatiti podatke s weba i formatirati HTML
- **Napravimo**:
  - **Model binding – povežimo HTML elemente s JS objektima (podatcima)**
  - **Automatske izračune ovisnih vrijednosti**
  - **Automatsko iscrtavanje (rendering)**



Povezivanje podataka (UI data binding)

- Obrazac kod razvoja GUI aplikacija
- Povezivanje elemenata GUI-ja i domenskog modela
  - Two-way (npr. Angular)
  - Unidirectional (npr. React)



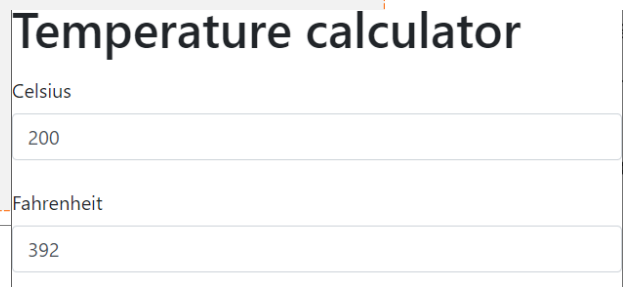
Pogledajmo prvo klijentsku stranu – Calculator

ver4

Calculator.js

```
export default class  {
  constructor () {
    this.myData = {
        tempCelsius: 0,
        tempFahrenheit: 32
    };
  }
  async getHtml() {(...)}
  getData() { return this.myData; }
  onChange(key, value) {
    if (key === "tempCelsius") {
      this.myData.tempFahrenheit = parseFloat(value) * 9 / 5 + 32;
    } else if (key === "tempFahrenheit") {
      this.myData.tempCelsius = (parseFloat(value) - 32) * 5/ 9;
    }
  }
}
```

```
async getHtml() {
  return `
    <h1>Temperature calculator</h1>
    <form>...
      <input type="number"
        id="celsius" placeholder="C"
        uspa-bind="tempCelsius">
      ...
      <input type="number"
        id="fahrenheit" placeholder="F"
        uspa-bind="tempFahrenheit">
      ...</form>`;
}
```



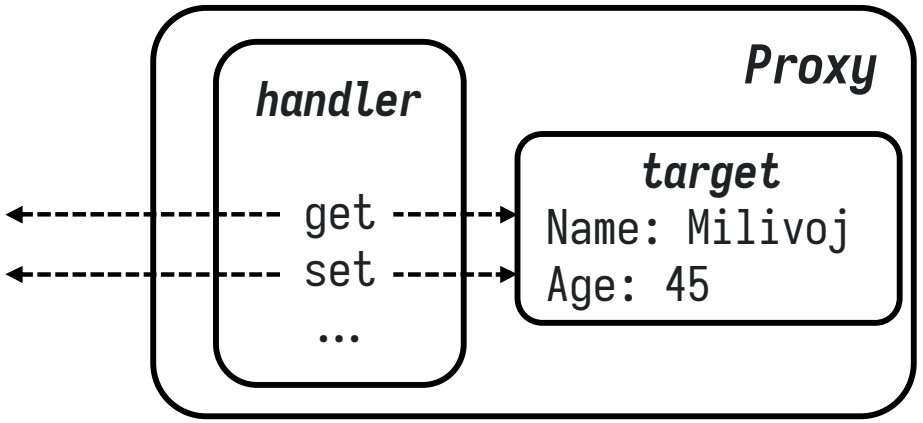
„Digresija”: Proxy

ver4

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Proxy

ProxyExample.js

```
let obj = {  name: "Milivoj",  age: 45 };
let pObj = new Proxy(obj, {
  get: function (target, prop) {
    console.log("Netko želi znati ", prop);
    return target[prop];
  },
  set: function (obj, prop, value) {
    console.log("Netko postavlja ", prop, "na", value);
    if (prop === "age") {
      if (!Number.isInteger(value)) {
        throw new TypeError("Godine moraju biti cijeli broj!");
      }
    }
    obj[prop] = value;
    return true; // Indicate success
  },
});
console.log(obj.name, obj.age);
console.log(pObj.name, pObj.age);
try {
    pObj.age = "33a";
} catch (error) {
    console.error(error);
}
pObj.age = 33;
console.log(obj);
```




```
Milivoj 45
Netko želi znati  name
Netko želi znati  age
Milivoj 45
Netko postavlja  age na 33a
TypeError: Godine moraju biti cijeli broj!
    at Object.set (...digressions\ProxyExample.js:15:15)
    (...)
    at Object.<anonymous> (...\digressions\ProxyExample.js:26:14)
Netko postavlja  age na 33
{ name: 'Milivoj', age: 33 }
```

ver4

Dodajmo i Observer obrazac

Observer.js

```
export default class Observer {
  constructor(dataObject, listener) {
    this.observersSet = [];
    if (listener) this.observersSet.push(listener);
    let self = this;
    this.proxyObject = new Proxy(dataObject, {
      set: (target, key, value, receiver) => {
        const result = Reflect.set(target, key, value, receiver);
        self.dataObjectClone = {...dataObject};  // used in isDirty()
        self.observersSet.forEach(observer => observer(key, value));
        return result;
      }
    });
  }
  isDirty() { // poor man's micro-optimization
    for (const key in this.dataObjectClone) {
      if (this.dataObjectClone[key] !== this.proxyObject[key]) { return true; }
    }
    return false;
  }
}
```

U našem slučaju, zainteresiran je:

`Calculator.onChange(key, value)`

ver4

Konačno – povežimo sve

uspa.js

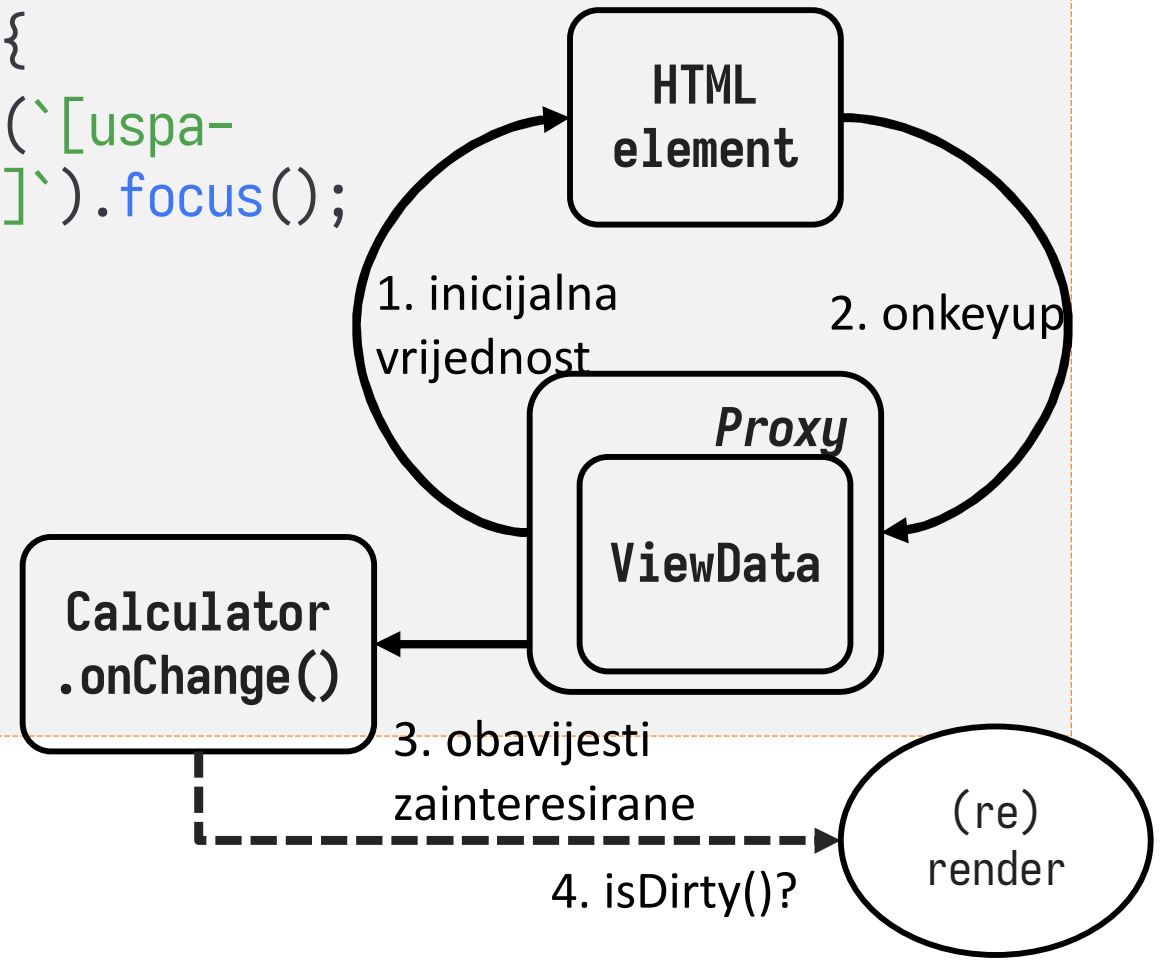
```
async setView(viewName) {
  let viewClass = this.currView.getViews()[viewName];
  this.currViewObject = new viewClass(); // should I cache it?
  await this.render();
  history.pushState(null, null, `${this.stubUrl}/${viewName}`);
  this.bindViewData(); }
bindViewData() {
  if (this.currViewObject.getData) {
    let currViewData = this.currViewObject.getData();
    this.currViewProxy = new Observer(currViewData,
                                      this.currViewObject.onChange.bind(this.currViewObject));
    document.querySelectorAll("[uspa-bind]").forEach((elem) => {
      let name = elem.getAttribute("uspa-bind");
      let proxy = this.currViewProxy.getProxy();
      elem.value = proxy[name];
      elem.onkeyup = () => {
        proxy[name] = elem.value;  // could use some metadata and do parsing, eg int, date, etc
        if (this.currViewProxy.isDirty()) {
          this.render(name);
        }
      };   });  } }
```

Izmjestili u posebnu metodu (sljedeći slajd)

...nastavak:

ver4

```
  async render(focusOnElementName) {  // silly rendering...
    document.querySelector("router-
view").innerHTML = await this.currViewObject.getHtml();
    this.bindViewData(); // ☹
    if (focusOnElementName) {
      document.querySelector(`[uspa-
bind="${focusOnElementName}"]`).focus();
    }
  }
```

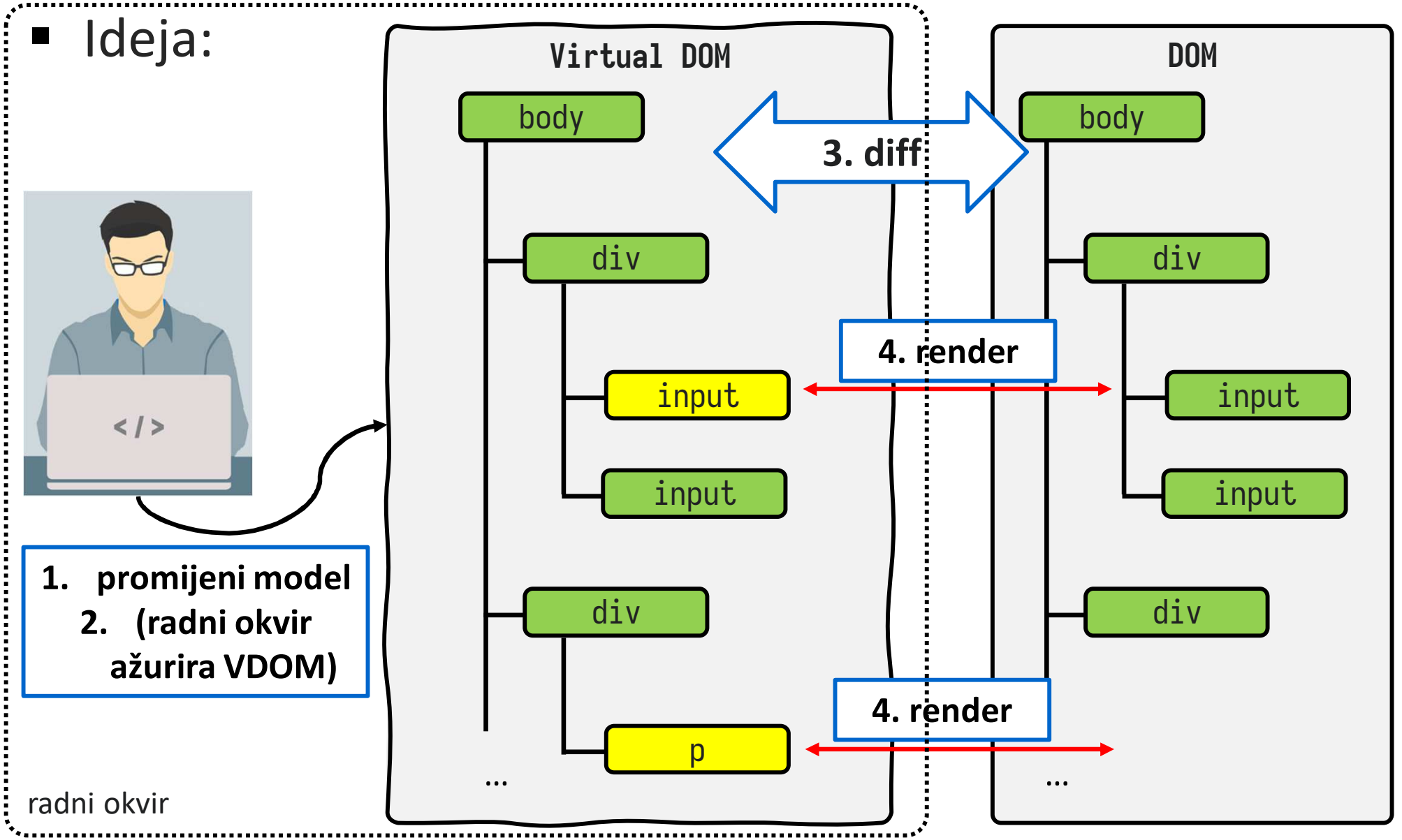


„Silly” rendering…

- Iscrtavanje na takav način bi bilo loše - kod svake promjene se:
  - ponovo iscrtava cijeli sadržaj
  - i ponovo se radi povezivanje svega!
- Različiti radni okviri koriste različite pristupa za optimiranje iscrtavanja, npr.:
  - React i Vue koriste tzv. Virtual DOM
  - Angular koristi „Incremental DOM” https://github.com/google/incremental-dom
  - Svelte ni jedno ni drugo, itd.
- S korisničke strane želi se postići da korisnik (developer) ne mora brinuti o (performansama) iscrtavanja, već da se bavi deklarativnim programiranjem i promjenama stanja

Virtual DOM

- Nezahtjevna JS memorijska reprezentacija DOM-a
- Ideja:



Kraj primjera

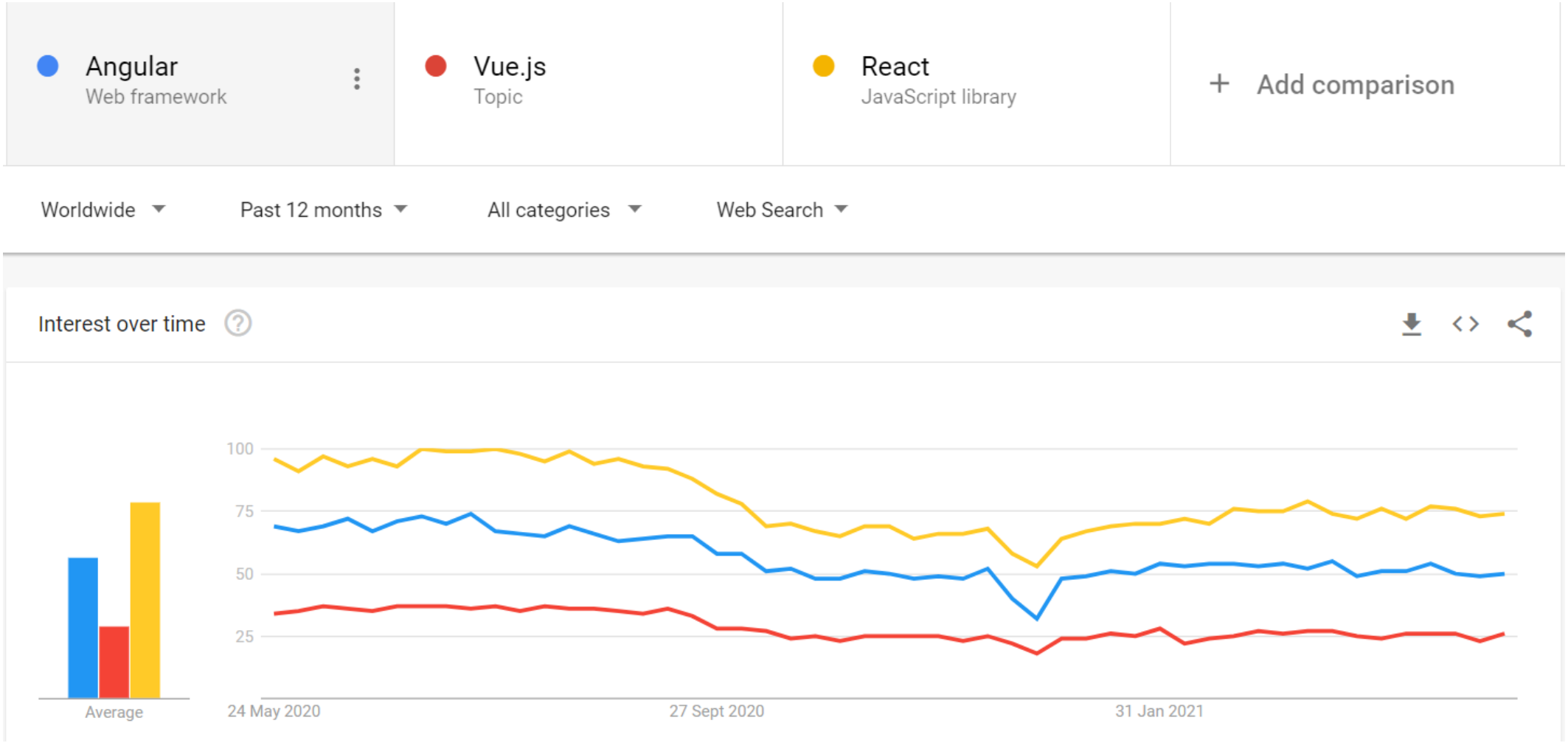
- Vidjeli smo:
- Routing
  - History API
- Data-binding
  - Proxy, Observer
- Rendering
  - Virtual DOM
- (HTML) templates
- „Components”

A sad idemo na pravi radni okvir…

Vue.js

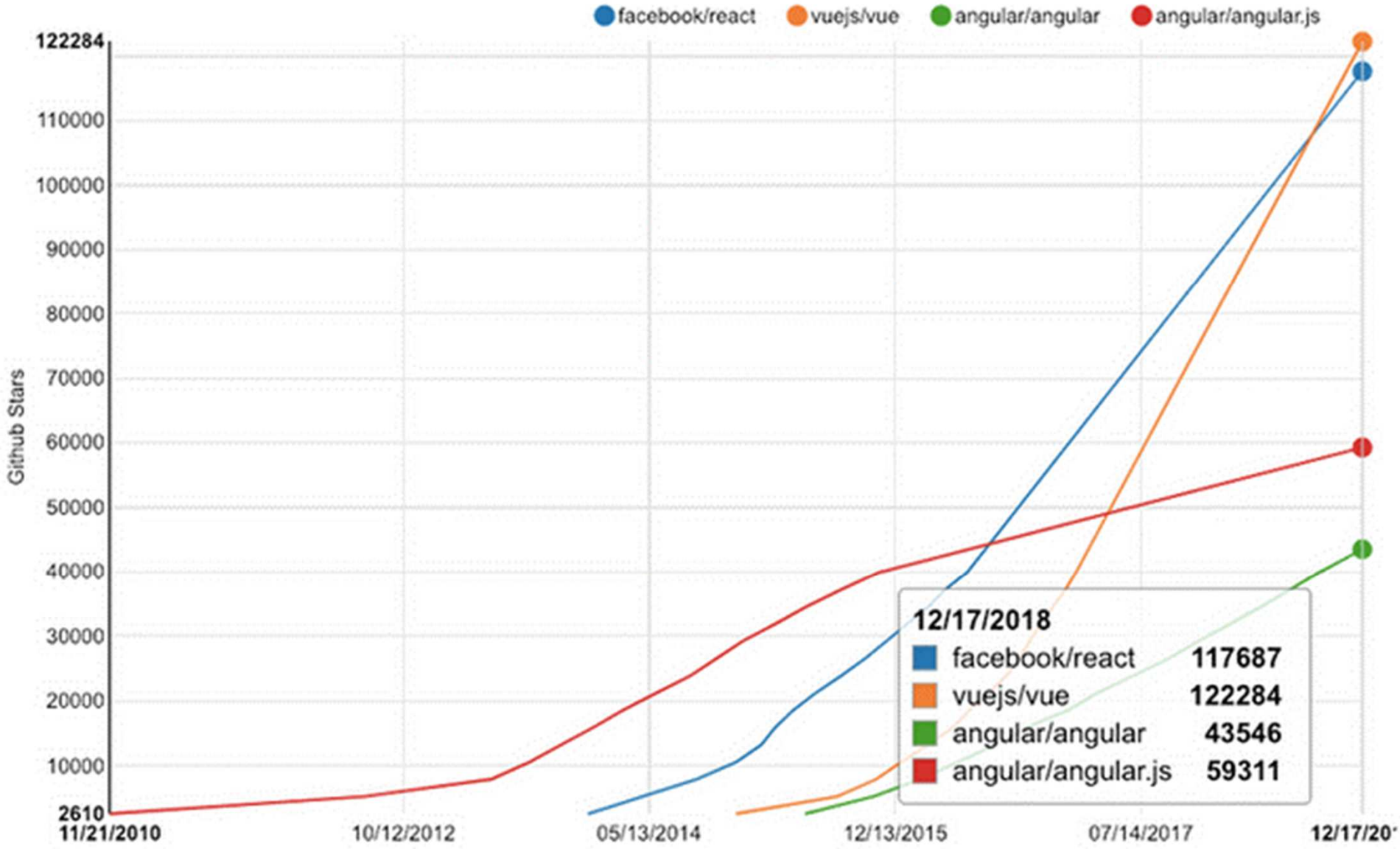
a crash course…

Velika trojka: Angular, Vue, React



https://trends.google.com/trends/explore?q=%2Fg%2F11c6w0ddw9,%2Fg%2F11c0vmgx5d,%2Fm%2F012l1vxv

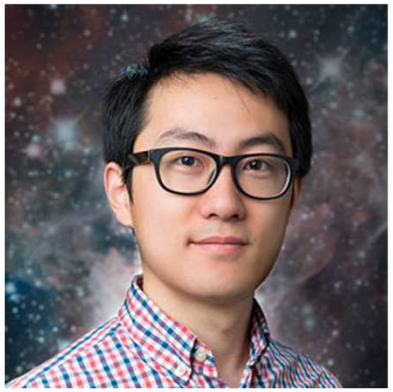
Github stars



https://theappsolutions.com/blog/development/vue-angular-react-comparison/

Vue.js

- „nije važno” -> sva tri su dobra
- Wikipedia:
  - Vue.js (commonly referred to as Vue; pronounced /vjuː/, like "view") is an **open-source model–view–viewmodel front end** JavaScript **framework** for building **user interfaces** and **single-page applications**.
- Vue.js:
  - Vue is a **progressive framework** for building **user interfaces**.



- Evan You, 2014.:
  - *"I figured, what if I could just extract the part that I really liked about **Angular** and build something really lightweight.„*

https://web.archive.org/web/20170603052649/https://betweenthewires.org/2016/11/03/evan-you/

- *Vue 2 -> **Vue 3***

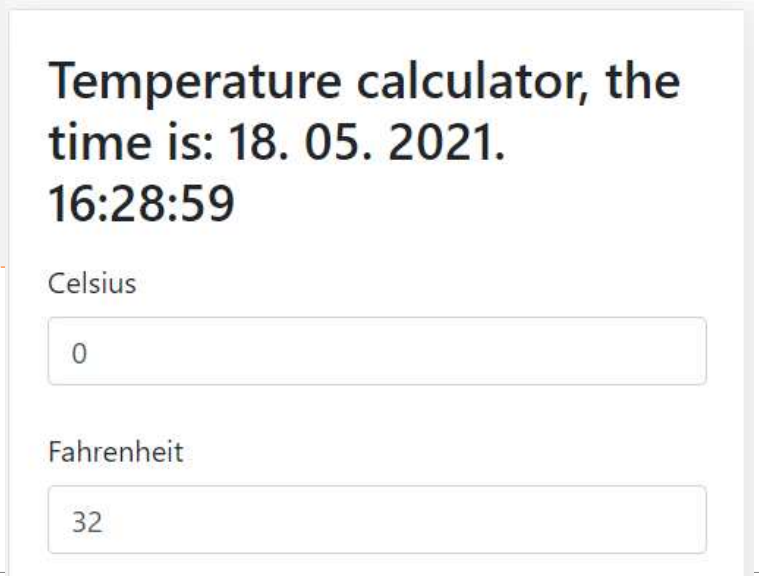
01-basics, interpolation, v-bind

Primjer – Calculator

app.js

```
const Calculator = {
  data() {
    return {
      tempCelsius: 0,
      tempFahrenheit: 32,
      startedDateAt: new Date().toLocaleDateString("hr-HR"),
      startedTimeAt: new Date().toLocaleTimeString("hr-HR"),
    };
  },
};
const app = Vue.createApp(Calculator);
app.mount("#my-app");
```

HTML (template) je zasad u index.html (sljedeći slajd)



Primjer – Calculator

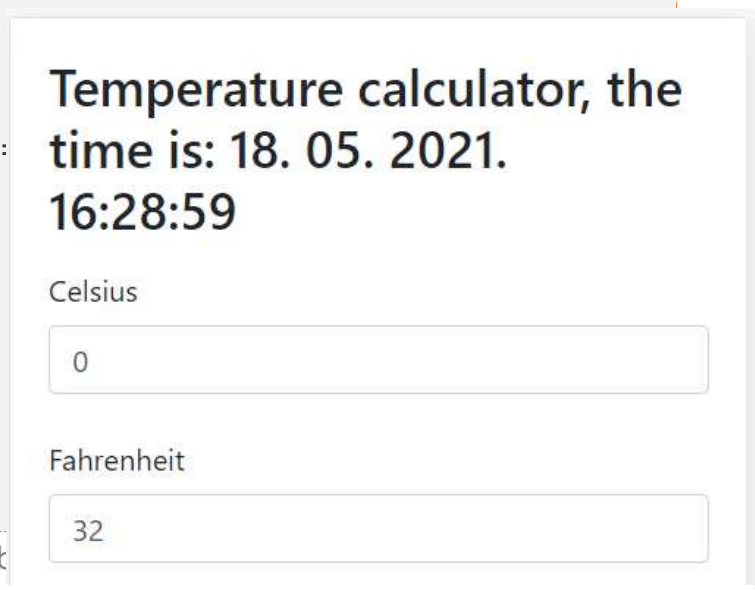
index.html

```
<body>
  <div id="my-app">
    <h3>Temperature calculator, the time is: {{ startedDateAt }} {{ startedTimeAt }} </h3>
    <form>
      <div class="form-group row">
        <label for="celsius" class="col-sm-2 col-form-label">Celsius</label>
        <div class="col-sm-10">
          <input type="number" class="form-control" placeholder="C"
            v-bind:value="tempCelsius">
        </div>
      </div>
      <div class="form-group row">
        <label for="fahrenheit" class="col-sm-2 col-form-label"
        <div class="col-sm-10">
          <input type="number" class="form-control" placeholder=
           :value="tempFahrenheit">
        </div>
      </div>
    </form>
  </div>
</body>
```

interpolation: {{ js }}

v-bind: jednosmjerno

v-bind: skraćena sintaksa



02-basics, events, methods, this

Dodajmo interakciju – prvo dodajem methods

app.js

```
const Calculator = {
  data() {
    return {
      tempCelsius: 0,
      tempFahrenheit: 32, (...)
    };
  },
  methods: {
    c2f() {
      this.tempCelsius = this.toFloat(this.$refs.tempCelsius.value);
      this.tempFahrenheit = Math.round(this.tempCelsius  * 9 / 5 + 32);
    },
    f2c(event) {
      event.preventDefault();
      this.tempFahrenheit = this.toFloat(this.$refs.tempFahrenheit.value);
      this.tempCelsius = Math.round((this.tempFahrenheit    32) * 5/ 9);
    },
    toFloat(value) {  // suvišno, samo da se vidi this.method
      return parseFloat(value);
    }
  }
};
```

Pored data, sad imamo i methods

Očigledno, svojstva podatkovnog objekta i funkcije u methods bivaju mapirane na this *(Proxy)*

Pozivat ćemo ih onclick; ako nas zanima Vue će nam poslati i event object.

Ugrađeni $refs objekt će sadržavati HTML elemente koje označimo (sljedeći slajd)

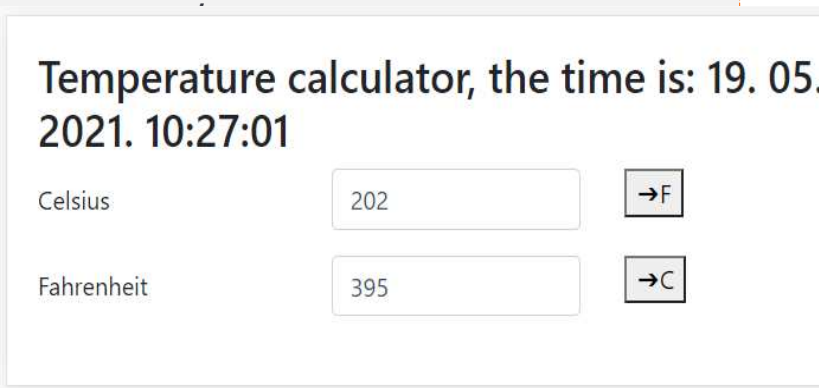
Dodajmo interakciju (pretvorbu)

index.html

```
<form>
  <div class="form-group row">
    <label for="celsius" class="col-sm-4 col-form-label">Celsius</label>
    <div class="col-4">
      <input type="number" class="form-control" placeholder="C"
        v-bind:value="tempCelsius" ref="tempCelsius">
    </div>
    <div class="col-2">
      <button v-on:click="c2f">➔F</button>
    </div>
  </div>
  <div class="form-group row">
    <label for="fahrenheit" class="col-4 col-form-label" >Fa
    <div class="col-4">
      <input type="number" class="form-control" placeholder=
        :value="tempFahrenheit" ref="tempFahrenheit">
    </div>
    <div class="col-2">
      <button @click.prevent="f2c">➔C</button>
    </div>
  </div>
</form>
```

dodajemo u $refs, ime je proizvoljno

v-on:event: method ili method(...)

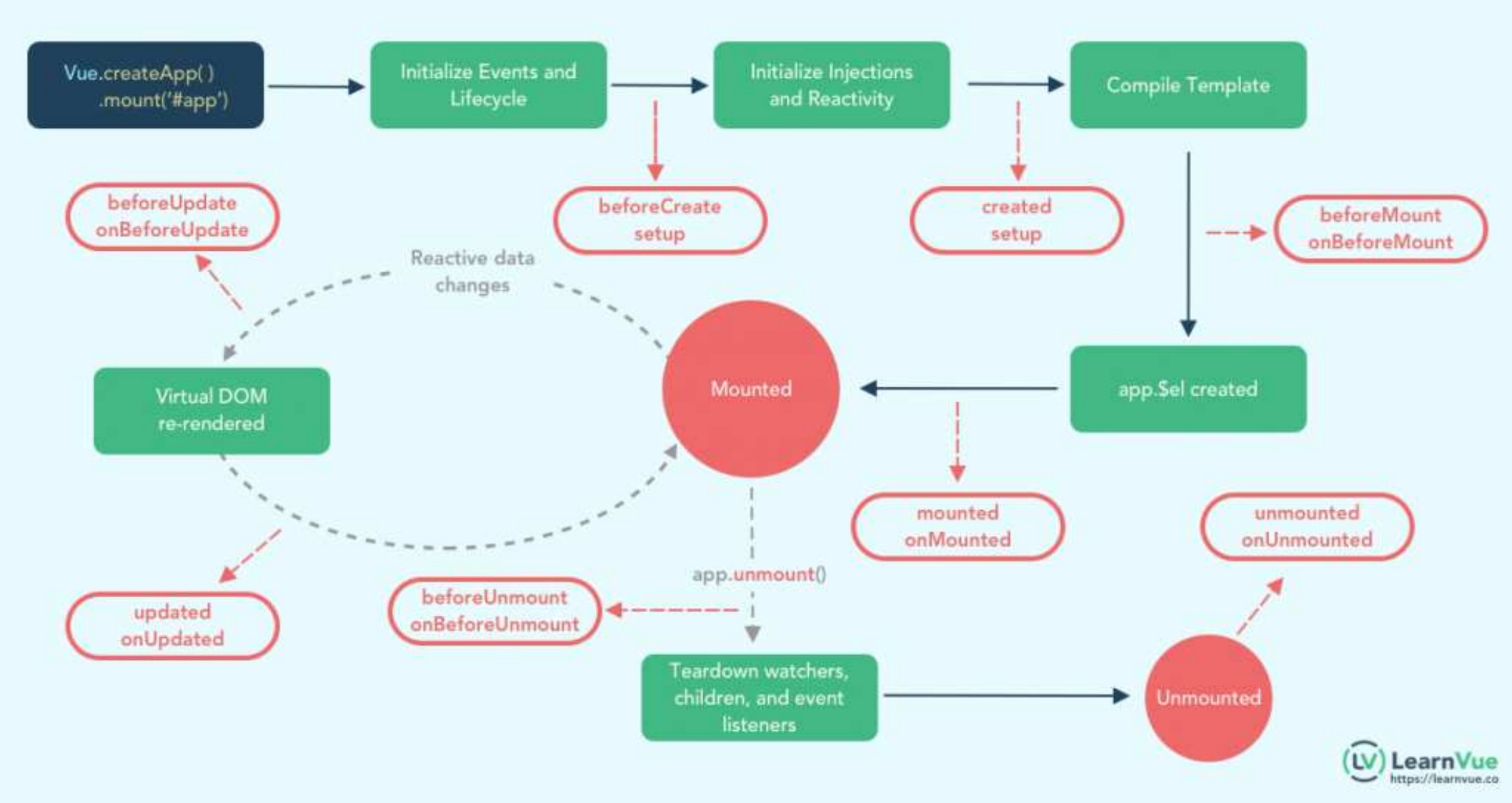


v-on: skraćena sintaksa @

event modifier – otkazuje submit

03-basics, v-model, lifecycle

Vue3 Lifecycle



https://learnvue.co/2020/12/how-to-use-lifecycle-hooks-in-vue3/

v-model: povezujemo UI i varijablu modela (i uklanjamo gumbe)

index.html

```
<h3>Temperature calculator, the time is: {{ startedDateAt }} {{ startedTimeAt }
} </h
<form
    <
    <label          elsius" class="col-sm-4 col-form-label">Celsiu
    <div c          "col-4">
        <in         ut type="number" class="form-control" placeholder="C"
        v-model="tempCelsius" @keyup="c2f">
    </div>
    </div>
    <div class="form-group row">
    <label for="fahrenheit" class="col-4 col-form-label" >Fahrenheit</label>
    <div class="col-4">
        <input type="number" class="form-control" placeholder="F"
        v-model.number="tempFahrenheit" @keyup.enter="f2c">
    </div>
    </div>
</form>
```

Two-way binding: v-model

Mijenjat ćemo svake sekunde! Što je s iscrtavanjem?

Kod svake promjene izračunavamo F

Binding modifier: vue će pretvarati string u number

keyup modifier: samo ako je enter

Lifecycle event - mounted

app.js

```
methods: {
  c2f() {
    this.tempFahrenheit = Math.round((this.tempCelsius * 9) / 5 + 32);
  },
  f2c() {
    console.log(typeof this.tempFahrenheit);
    this.tempCelsius = Math.round(((this.tempFahrenheit - 32) * 5) / 9);
  }
},
mounted() {
  window.setInterval(() => {
    this.startedTimeAt = new Date().toLocaleTimeString("hr-HR");
  }, 1000);
}
```

Primijetiti da više ne moramo postavljati C, lokalna varijabla `this.tempCelsius` je automatski ažurna

Zbog binding modifera ovo će biti number

Lifecycle event: https://v3.vuejs.org/guide/instance.html#lifecycle-diagram

Svaku sekundu ažuriramo vrijednost jedne varijable modela i ne brinemo za iscrtavanje!

04-basics: v-for, v-if, :class, template

Dodajmo novu varijablu i dvije funkcije

app.js

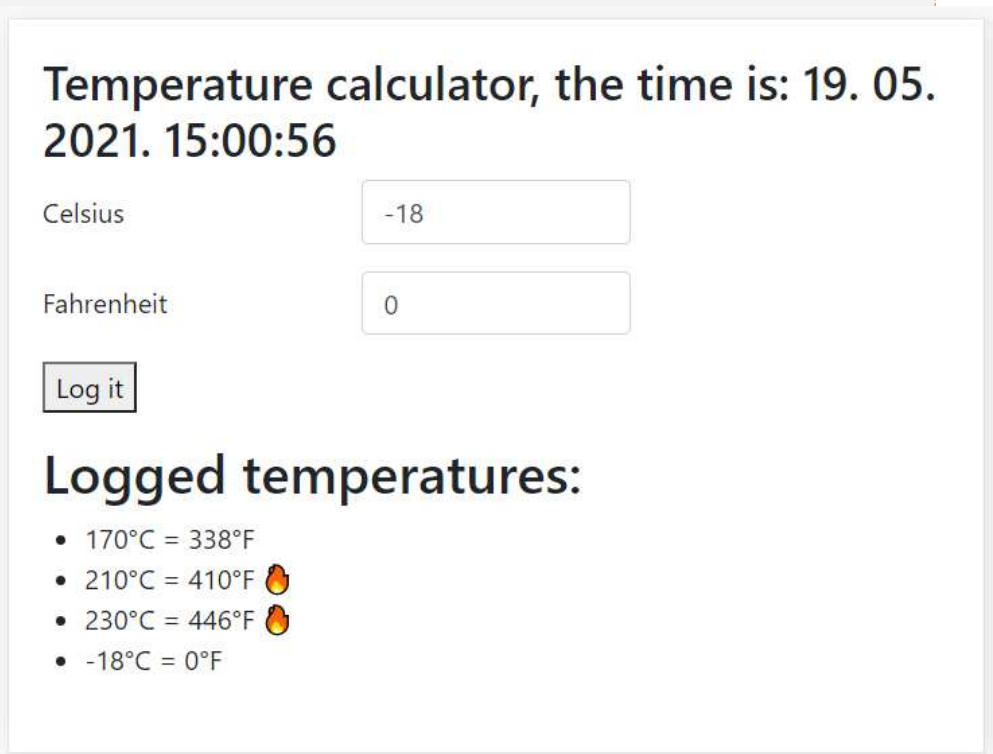
```
...
data() {
  return {
    (...)
    logItems: []
  };
},
methods: {
  c2f() { ... },
  f2c() { ... },
  logItemClass(item) {
     return  (item.C > 200) ? "hot" : "";
   },
   logTemp() {
     this.logItems.push({
       C: this.tempCelsius,
       F: this.tempFahrenheit
     })
   }
},
```

Dodajmo polje

Dodajemo trenutne vrijednosti temperatura u polje

styles.css

```
...
.hot::after {
    content: " \1F525";
    color: red;
    font-weight: bold;
}
```



Preselili HTML iz index.html -> app.js

app.js

```
const Calculator = {
  template: `<h3>Temperature calculator, the time is: {{starte
  <form>
      (...)
    <div class="form-group row">
        <div class="col-12">
            <button type="button" @click.prevent="logTemp">Log it</button>
        </div>
    </div>
    <div v-if="logItems.length > 0">
      <h2>Logged temperatures:</h2>
      <div class="form-group row">
          <ul>
            <li v-for="item in logItems" :class="logItemClass(item)">
                {{ item.C }}°C = {{ item.F }}°F
            </li>
          </ul>
      </div>
    </div>
</form> `,
  data() {...
```

index.html

```
<body>
    <div id="my-app">
    </div>
</body>
```

Izgubili smo *syntax coloring*, vidjet ćemo kasnije bolji način za definirati template unutar „komponente”

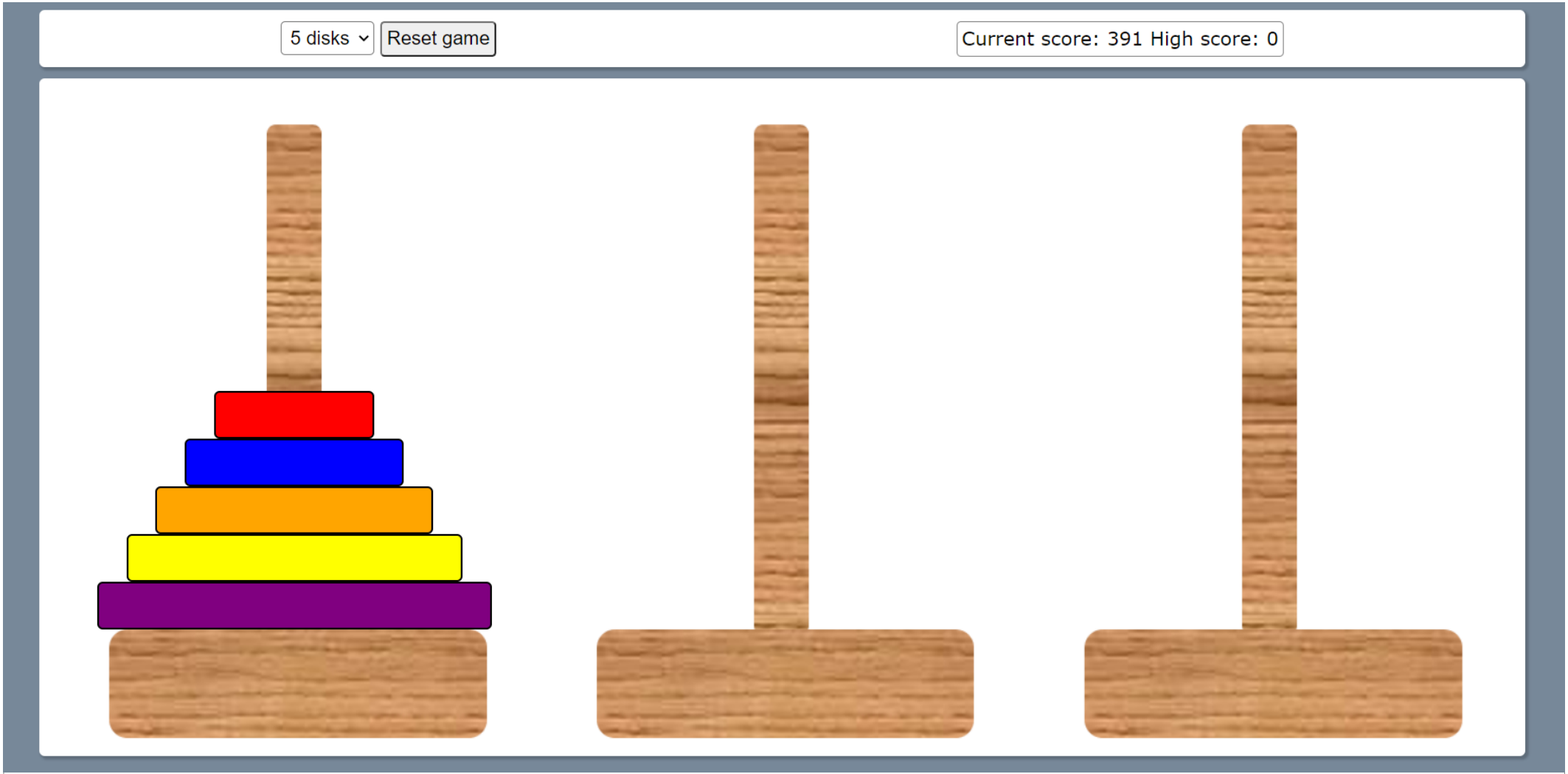
Ako je polje prazno, nema cijelog div bloka

Iteriramo po elementima polja, postoji i sintaksa za dobiti indekse

Bind klase (class), metoda će vratiti ime. Nije u koliziji s eventualno ručno postavljenim klasama.

Novi primjer – Hanojski tornjevi

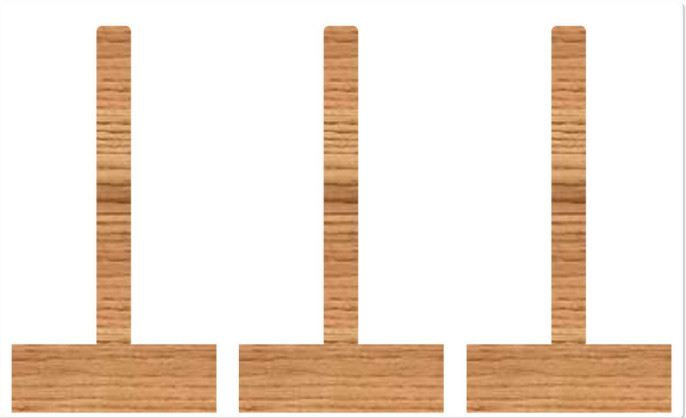
https://en.wikipedia.org/wiki/Tower_of_Hanoi



Inicijalne postavke – HTML, CSS

index.html

```
...
<div id="app">
    <div class="board">
        <div class="rod">
        </div>
        <div class="rod">
        </div>
        <div class="rod">
        </div>
    </div>
</div>
...
```



hanoi.css

```
div.board {
  margin: auto;
  width: 1000px;
  border-radius: 5px;
  box-shadow: 2px 2px 2px 1px rgba(0, 0, 0, 0.2);
  display: flex;
  justify-content: space-around;
}
div.rod {
  width: 300px;
  height: 500px;
  background-image: url("./stick.png");
  background-position: center;
  background-repeat: no-repeat;
  background-size: cover;
  display: flex;
  align-items: center;
  flex-direction: column;
  justify-content: flex-end;
  padding-bottom: 107px;
}
div.disk {
  height: 40px;
  border-radius: 5px;
  border: 2px solid black;
}
```

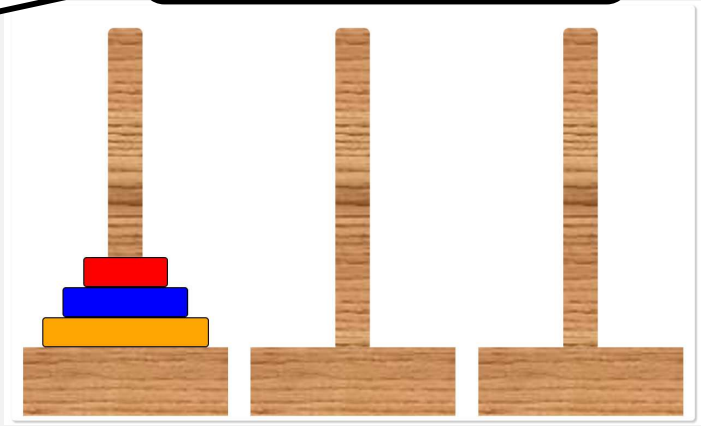
v-for, style – iscrtavamo N diskova

index.html

```
...
<div class="board">
  <div class="rod">
    <div v-for="disk in positionA"
      class="disk"
      @click="onSelectFrom(disk)"
      :style="rodStyle(disk)">
    </div>
  </div>
  <div class="rod">
    <div v-for="disk in positionB"
      class="disk"
      @click="onSelectFrom(disk)"
      :style="rodStyle(disk)">
    </div>
  </div>
  <div class="rod">
    <div v-for="disk in positionC"
      class="disk"
      @click="onSelectFrom(disk)"
      :style="rodStyle(disk)">
    </div>
  </div>
</div>
...
```

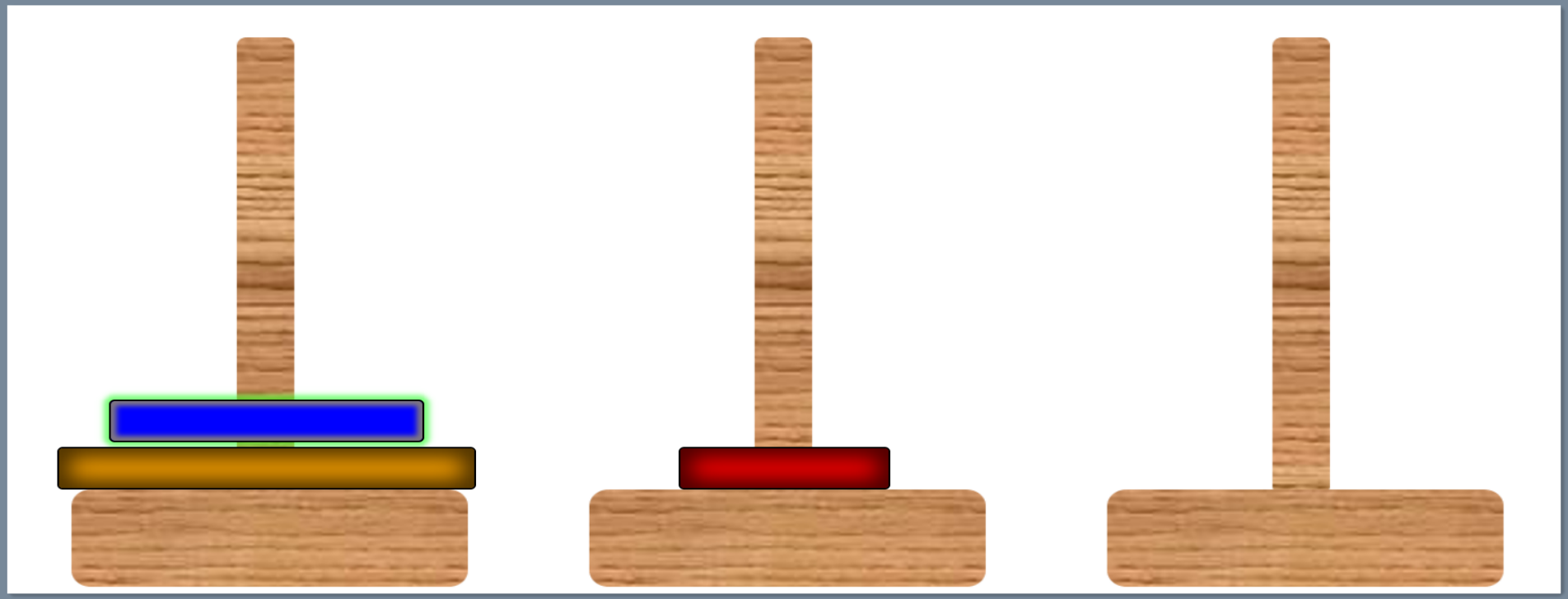
hanoi.js

Nova sintaksa



```
var app = new Vue({
  el: '#app',
  data: {
    diskNumber: 3,
    positionA: [1, 2, 3],
    positionB: [],
    positionC: [],
  },
  methods: {
    rodStyle(diskSize){
      return {
        width: (diskSize / this.diskNumber * 60 + 20) + '%',
        backgroundColor: this.getRodColor(diskSize)
      }
    },
    getRodColor(diskSize){
      const colors = ['green', 'red', 'blue',
          'orange', 'yellow', 'purple'];
      return colors[diskSize % colors.length];
    }
  }
})
```

Omogućimo vizualno isticanje odabranog diska, prebacivanje diskova (uz pravila), te „dnevnik”



21. 05. 2021. 11:11:56	No disk may be placed on top of a disk that is smaller than it.
21. 05. 2021. 11:04:48	Selected disk: 2 at rod A
21. 05. 2021. 11:04:47	Moved disk 1 from A to B
21. 05. 2021. 11:04:45	Selected disk: 1 at rod A

Congrats, game over in 7 moves! **That is optimal.**

21. 05. 2021. 11:14:00	Moved disk 1 from A to B
21. 05. 2021. 11:13:59	Selected disk: 1 at rod A
21. 05. 2021. 11:13:58	Moved disk 2 from C to B
21. 05. 2021. 11:13:57	Selected disk: 2 at rod C
21. 05. 2021. 11:13:56	Moved disk 1 from C to A
21. 05. 2021. 11:13:55	Selected disk: 1 at rod C
21. 05. 2021. 11:13:54	Moved disk 3 from A to B
21. 05. 2021. 11:13:53	Selected disk: 3 at rod A
21. 05. 2021. 11:13:52	Moved disk 1 from B to C
21. 05. 2021. 11:13:51	Selected disk: 1 at rod B

Reactivity

index.html

```
...
<div v-if="isGameOver" class="board card">
    <h1>Congrats, game over in {{ moves }} moves!</h1>
    <h2>That is {{ moves !== (Math.pow(2, diskNumber)-1) ? "SUB" : "" }}optimal.</h2>
</div>
<div v-else class="board card">
  <div class="rod" @click="onSelectTo('A')">
    <div v-for="disk in positionA"
       class="disk"
       :class="{selectedDisk : disk === selectedDisk, inactiveDisk: selectedDisk != null && disk !== selectedDisk}"
       @click="onSelectFrom(disk, 'A')"
       :style="rodStyle(disk)">
    </div>
  </div>
Analogno za druga dva...
```

`v-if` je igra gotova, ispišemo je li optimalan broj poteza,
`v-else` prikazujemo igru (board)

(uvjetno) postavljamo klase selectedDisk i inactiveDisk

DM5
IM1

hanoi.js

```
var app = new Vue({
  el: '#app',
  data: { diskNumber: 3,
    positionA: [1, 2, 3],   positionB: [],     positionC: [],
    selectedDisk: null,     selectedRod: null,
    logMessages: [],
    moves: 0
  },
  computed: {
    isGameOver: function() {
      return this.positionB.length === this.diskNumber
        || this.positionC.length === this.diskNumber
    }
  }, ... nastavak na sljedećem slajdu...
```

`computed` property – kada imamo iole složeniju logiku u templateima (koja se temelji na modelu) možemo ju premjestiti u computed property. Npr. izračun optimalnog broja poteza gore se također mogao „izmjestiti”

Slide 34

DM5 :class se može računati u metodi, ovdje je samo radi primjera...
Danijel Mlinarić; 13.9.2021.

IM1 zapravo prije u computed prop, a ne u metodi.
Igor Mekterović; 2.12.2021.

hanoi-03

Reactivity: odabir i spuštanje diska

hanoi.js

```
methods: { ...
  onSelectFrom(disk, rod) {
      if (
      (this.positionA[0] === disk)
      || (this.positionB[0] === disk)
      || (this.positionC[0] === disk)
      ) {
      this.selectedDisk = disk;
      this.selectedRod = rod;
      this.log("Selected disk: "
     + this.selectedDisk + " at rod "
       + this.selectedRod);
      } else {
          this.log("You can only select the
topmost disk.");
      }
  },
  log(msg) {
     this.logMessages.unshift({
     ts: new Date().toLocaleString("hr-HR"),
     text: msg,
   });
  },
...
```

hanoi.js

```
onSelectTo(rod) {
    if (this.selectedDisk
         && rod !== this.selectedRod) {
    let positionABC = [this.positionA, this.positionB,
                       this.positionC];
    let idxTo = rod.charCodeAt(0) - "A".charCodeAt(0);
    let rodToArray = positionABC[idxTo];
    let idxFrom = this.selectedRod.charCodeAt(0)-"A".charCodeAt(0);
    let rodFromArray = positionABC[idxFrom];

    if (rodToArray.length && rodToArray[0] < this.selectedDisk) {
        this.log(
   "No disk may be placed on top of a disk that is smaller than it."
        );
    } else {
        rodFromArray.shift();
        rodToArray.unshift(this.selectedDisk);
        this.log("Moved disk " + this.selectedDisk + " from " +
            this.selectedRod + " to " +  rod
        );
        this.selectedRod = null;
        this.selectedDisk = null;
        this.moves++;
    }
  }
}
```

Slide 35

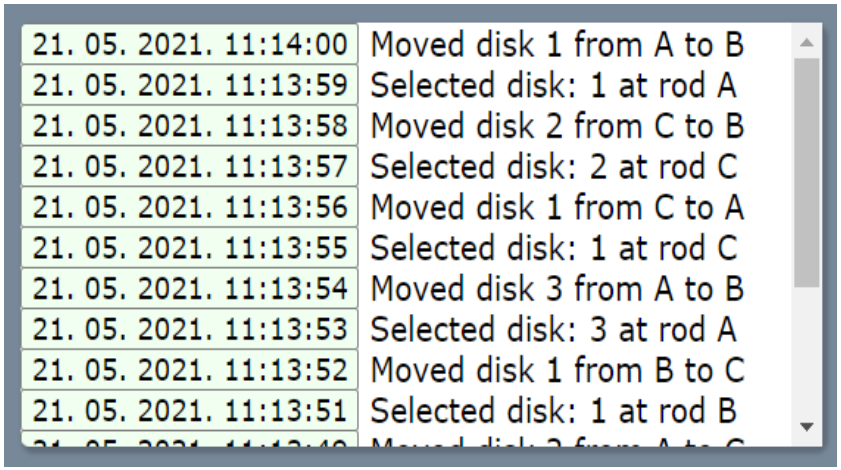
DM6 umjesto rod.charCodeAt(0) - "A".charCodeAt(0) jednostavno bih kao paramatera predao index
Danijel Mlinarić; 13.9.2021.

IM3 moglo bi se, ali onda je u kodu ručnije u templateu i važnije - kod ispisa u logu moram ratidi obratno - morao bih pretvarati 0 u A, 1 u B ... tako da je slično...
Igor Mekterović; 2.12.2021.

Ispis dnevnika i CSS

index.html

```
...
<div class="log card">
    <div v-for="msg in logMessages" >
      <span class="log--time">
        {{ msg.ts }}
      </span>
      <span class="log--message">
         {{ msg.text }}
      </span>
    </div>
</div>
```



hanoi.css

```
.selectedDisk {
  margin: 5px;
  box-shadow: 0px 0px 5px 5px rgba(0, 255, 0, 0.5),
  0px 0px 5px 5px inset rgba(255, 238, 0, 0.5);
}
.inactiveDisk {
  box-shadow: 0px 0px 20px 10px inset rgba(0, 0, 0, 0.7);
}
div.log {
  margin-top: 20px;
  max-height: 200px;
  overflow: auto;
  min-height: 100px;
  font-family: Verdana, Geneva, Tahoma, sans-serif;
}
.log--time {
  font-size: 0.9em;
  background-color: honeydew;
  padding: 0 5px 0 5px;
  border: 1px solid gray;
  border-radius: 2px;
}
```

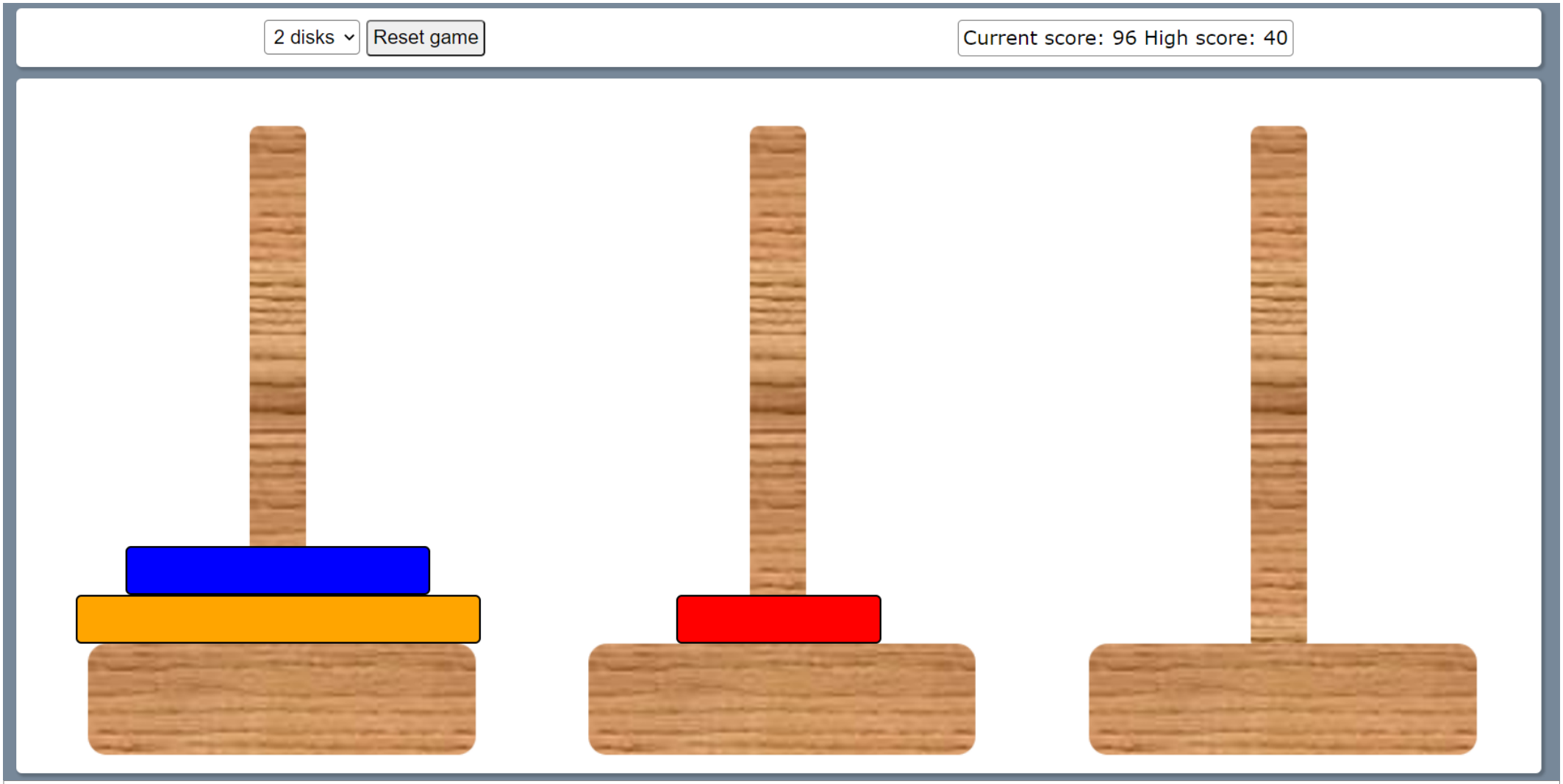
Par napomena uz *computed property*

- Sličnost s `methods`
  - Kada koristiti jedno, kada drugo?
    - **Computed properties** - kada samo mijenjamo prezentaciju, ali ne i podatke
      - Štoviše - **paziti da ne mijenjamo podatke** unutra *computed properties* - nezgodni bugovi!
        pogledajte: https://vueschool.io/lessons/computed-properties-in-vue-3?friend=vuejs
    - **Methods** - kada mijenjamo podatke
  - Bitna razlika - CP su **keširani**!
    - Ponovo se izračunavaju samo ako se promijeni neka **reaktivna** varijabla na temelju koje se izračunava
    - Npr. nikad se neće osvježiti:
    - (moguće isključiti cache)
- Postoji i Computed setter

```
computed: {
  now() {
    return Date.now()
  }
}
```

Omogućimo novu igru s različitim brojem diskova, izračunajmo score i zapamtimo ga kako se ne bi izgubio nakon osvježavanja stranice

hanoi-04



Dodajemo highScore, učitavamo, snimamo…

hanoi-04

index.html

```
<div class="board card">
  <div class="reset-game">
  <select ref="disksNumber"
      class="form-control">
      Number of disks:
      <option value="2">2 disks</option>
      <option value="3">3 disks</option>
      <option value="4">4 disks</option>
      <option value="5">5 disks</option>
  </select>
  <button @click="resetGame"
      class="form-control">Reset game
  </button>
  </div>
    <div class="high-score"
     :class="{ newHighScore: isNewHighScore }">
    Current score: {{score}}
    High score: {{highScore}}
    </div>
</div>
```

hanoi.js

```
computed: { …
  score: function () {
    return this.diskNumber * 10 -
       11 * (this.moves - this.optimalMoves());
  },
  isNewHighScore: function () {
    return this.isGameOver
       && (this.score > this.highScore);
    }
}, methods: { …
onSelectTo(rod) {
    …
    this.moves++;
    if (this.isNewHighScore) {
      this.highScore = this.score;
      localStorage.setItem('highScore', this.highScore);
      this.log("** CONGRATS!!! NEW HIGH SCORE!! **");
    }
  }
},
beforeMount() {
  let highScore = localStorage.getItem('highScore');
  if (highScore != null) {
    this.highScore = parseInt(highScore);
} …
```

Lifecycle event