HTML/CSS/JavaScript

ECMAScript 2016/ES7

Blick in die Zukunft

Kaum ist der ES2015-Standard verabschiedet, macht sich die die JavaScript-Entwicklergemeinde Gedanken über ES2016.

Listing 1: Promises

```
function fetch (url) {
  return new Promise(function (resolve, reject) {
    let request = new XMLHttpRequest();
    request.open("GET", url);
    request.responseType = 'json';

    request.onload = function () {
      if (request.status === 200) {
        resolve(request.response);
      } else {
        reject(Error(`XHR did fail:
${request.statusCode}`));
      }
    }

    request.onerror = function () {
      reject(Error("Network error."))
    }
  }
}

function getUser (name) {
  return fetch(`/users/${name}`);
}

function getGroup (name) {
  return fetch(`/groups/${name}`);
}

function showProfile (username, groupname) {
  getUser(username).then(user => {
    getGroup(groupname)])
      .then([user, group]) =>  {
        render(user,group);
      })
      .catch(e => showError(e))
  }).catch(e => showError(e))
}
```

Listing 2: Observer

```
function interval (period) {
 return new Observable(observer => {
  let count = 0,
    timeout,
    handle = function () {
      observer.next(count);
      count = count + 1;
      timeout = setTimeout(_=> {
        handle();
      }, period);
     }
  handle();
  return _=> {
    clearTimeout(timeout);
    observer.complete();
  }
 });
```

```
}
let timeInterval = interval(1000); // Jede Sekunde
```

Listing 3: Observable-Stream aus HTML-DOM

```
function fromEvent (element, eventName) {
  return new Observable(observer=> {
    function handler (event) {
      observer.next(event);
    }
    element.addEventListener(eventName, handler);
    return _=> element.removeEventListener(eventName,
handler);
  })
}

let moveevents = fromEvent(document, "mousemove");
let clicks =
Observable.fromEvent(document.getElementById("#button"),
"click");
```

Listing 4: Cycle und RxJS-Observables

```
/** @jsx hJSX */

import Cycle from '@cycle/core';
import {hJSX, makeDOMDriver} from '@cycle/dom';

function model(actions) {
  return actions.startWith(0).scan((x,y)=>x+y);
}

function intent(DOM) {
  return DOM.get("#increment", "click").map(ev=>+1)
}

function view(state) {
  return state.map(counter =>
    <div>
      <button id="increment">Increment</button>
      <p>Counter: {counter}</p>
    </div>
  );
}

function main({DOM}) {
  let actions = intent(DOM),
    state = model(actions);
  return {
    DOM: view(state)
  }
}

Cycle.run(main, {
  DOM: makeDOMDriver('#app')
});
```

Listing 5: Class Decorators

```
function meinKlassenDekorator(target) {
  console.log(`Konstruktor: ${target.name}`)
}

function meinDekorator (target, name, descriptor) {
  console.log(`Target: ${target.constructor.name}`);
  console.log(`Name: ${name}`);
  console.log(`Descriptor: ${JSON.stringify(descriptor)}`);
}

@meinKlassenDekorator
```

```
class MeineKlasse {
  @meinDekorator
  meineMethode () {
  }
}
```

Listing 6: Object.observe

```
var myObj = {
    name: "Hans Muster",
    beruf: "Friseur"
};

Object.observe(myObj, function (changes) {
    changes.forEach(function (change) {
        console.log("Property: "+change.name);
        console.log("New Object:
"+JSON.stringify(change.object));
        console.log("Change Type: "+change.type);
    })
});

myObj.name = "Fritz Müller"
// Ausgabe
Property: name
New Object: {"name":"Fritz Müller", "beruf":"Friseur"}
Change Type: update
```