


Asynchronität mit Promises

Alex Thalhammer

Inhalt

- Http-Zugriff
- DEMO: Erste Lösung
- Promises
- DEMO: Bessere Lösung

A vintage black rotary telephone is the central focus, resting on a wooden desk. The phone's handset is on the left, and the base is on the right. The rotary dial is clearly visible, showing numbers 1 through 0. In the foreground, a black fountain pen with gold-colored accents lies horizontally on a piece of paper with handwritten notes. The background is a warm, yellowish glow, suggesting a lamp. A semi-transparent dark rectangle is overlaid on the middle of the image, containing the title text in white.

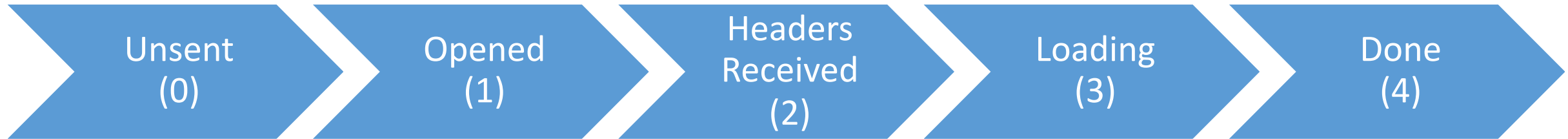
HTTP-Zugriff als Beispiel für asynchrone Operationen



XHR

XmlHttpRequest
onreadystatechange readyState responseText responseXML status statusText
open() send() abort() setRequestHeaders() getResponseHeaders()

Readystate



XHR

```
function loadData() {  
    var xmlhttp;  
    xmlhttp = new XMLHttpRequest();  
  
    [...]  
}
```

XHR

```
function loadData() {  
    var xmlhttp;  
    xmlhttp = new XMLHttpRequest();  
  
    xmlhttp.onreadystatechange = function() {  
        if (this.readyState == 4 && this.status == 200) {  
            console.debug(this.responseText);  
        }  
    };  
  
    [...]  
}
```

XHR

```
function loadData() {  
    var xmlhttp;  
    xmlhttp = new XMLHttpRequest();  
  
    xmlhttp.onreadystatechange = function() {  
        if (this.readyState == 4 && this.status == 200) {  
            console.debug(this.responseText);  
        }  
    };  
  
    xmlhttp.open("GET", "http://...", true);  
    xmlhttp.send();  
}
```


DEMO



Promises

Promises

- Definiert einen Wert, der (noch) nicht zwingend bekannt ist.
- Zustände:
 - pending, fulfilled, rejected

Promises

```
function loadData() {  
    return new Promise((resolve, reject) => {  
        //Async:  
        resolve(4711);  
        //reject("err!");  
    });  
}
```

Promises

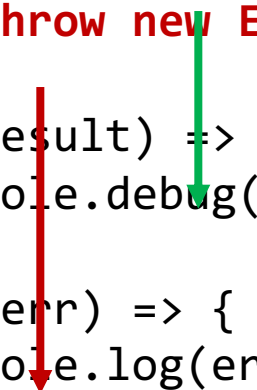
```
function loadData() {  
    return new Promise((resolve, reject) => {  
        //Async:  
        resolve(4711);  
        //reject("err!");  
    });  
}
```

Wird sofort ausgeführt!

```
loadData()  
    .then((result) => {  
        console.log(result);  
    })  
    .catch((err) => {  
        console.log(err);  
    });
```

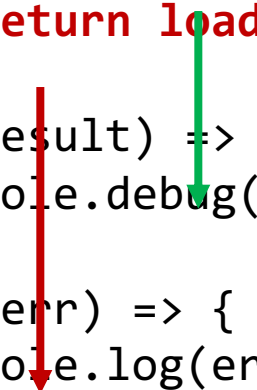
Promise Chaining

```
loadData()  
  .then((result) => {  
    return 'XYZ';  
    // throw new Error();  
  })  
  .then((result) => {  
    console.debug(result == 'XYZ');  
  })  
  .catch((err) => {  
    console.log(err);  
  });
```



Promise Chaining

```
loadData()  
  .then((result) => {  
    return loadData(...); // erfolgreich  
    // return loadData(...); // nicht erfolgreich  
  })  
  .then((result) => {  
    console.debug(result == 'XYZ');  
  })  
  .catch((err) => {  
    console.log(err);  
  });
```



DEMO