27th January 2020

TextDecoder and TextEncoder

What if the binary data is actually a string? For instance, we received a file with textual data.

The build-in TextDecoder object allows to read the value into an actual JavaScript string, given the buffer and the encoding.

We first need to create it:

```
let decoder = new TextDecoder([label], [options]);
```

- **label** – the encoding, `utf-8` by default, but `big5`, `windows-1251` and many other are also supported.
- **options** – optional object:
  - **fatal** – boolean, if `true` then throw an exception for invalid (non-decodable) characters, otherwise (default) replace them with character `\uFFFD`.
  - **ignoreBOM** – boolean, if `true` then ignore BOM (an optional byte-order unicode mark), rarely needed.

…And then decode:

```
let str = decoder.decode([input], [options]);
```

- **input** – `BufferSource` to decode.
- **options** – optional object:
  - **stream** – true for decoding streams, when `decoder` is called repeatedly with incoming chunks of data. In that case a multi-byte character may occasionally split between chunks. This options tells `TextDecoder` to memorize "unfinished" characters and decode them when the next chunk comes.

For instance:

```
let uint8Array = new Uint8Array([72, 101, 108, 108, 111]);

alert( new TextDecoder().decode(uint8Array) ); // Hello
```

```
let uint8Array = new Uint8Array([228, 189, 160, 229, 165, 189]);

alert( new TextDecoder().decode(uint8Array) ); // 你好
```

We can decode a part of the buffer by creating a subarray view for it:

```
let uint8Array = new Uint8Array([0, 72, 101, 108, 108, 111, 0]);

// the string is in the middle
// create a new view over it, without copying anything
let binaryString = uint8Array.subarray(1, -1);

alert( new TextDecoder().decode(binaryString) ); // Hello
```

TextEncoder

TextEncoder does the reverse thing – converts a string into bytes.

The syntax is:

```
let encoder = new TextEncoder();
```

The only encoding it supports is “utf-8”.

It has two methods:

- **`encode(str)`** – returns `Uint8Array` from a string.
- **`encodeInto(str, destination)`** – encodes `str` into `destination` that must be `Uint8Array`.

```
let encoder = new TextEncoder();

let uint8Array = encoder.encode("Hello");
alert(uint8Array); // 72,101,108,108,111
```

Previous lesson Next lesson

Share

Tutorial map

Comments

- If you have suggestions what to improve - please submit a GitHub issue or a pull request instead of commenting.
- If you can't understand something in the article – please elaborate.
- To insert a few words of code, use the `<code>` tag, for several lines – use `<pre>`, for more than 10 lines – use a sandbox (plnkr, JSBin, codepen…)

about the projectcontact usterms of usage privacy policy