JavaScript

ES6…

ECMA Script?

- Šta je uopšte ECMA Script?
  - Specifikacija skriptnog jezika
  - Kreirana je sa ciljem da se standardizuje implementacija JavaScript-a (mada ima i drugih implementacija)
  - Specifikacija je zaštićena i održava je Ecma International kao ECMA-262 i ISO/IEC 16262.

ES6?

- ECMAScript 6 (zvanično ECMAScript 2015)
- Predstavljao je značajno ažuriranje ECMAScript specifikacije nakon dosta godina ES6 (ES5 je zvanično izašao 2009, sa manjim naknadnim izmenama, dok su prethodne verzije ponekad čak i napuštane).
- Implementacija ovih specifikacija omogućila je bolju usklađenost JavaScript implementacija između različitih platformi, otvorila put upotrebi JavaScripta i van klijentskog okruženja.
- Nakon ES6 ažuriranje standarda se radi na godišnjem nivou, ali izmene su više evolutivne nego značajna proširenja jezika (kakvo je ES6)

Ključna proširenja ES6 - opseg važenja

- Uvedena podrška za block-scope varijable i konstante
  - dve nove ključne reči **let** i **const**
- Pre ES6 u JavaScriptu je postojao samo global i function scope
- Uvođenje blok opsega važenja rešava i problem redeklaracije varijable u različitim blokovima koda što je znao biti problem sa var

```
{
  var x = 2;
}

// ali x je vidljivo i ovde nakon kraja bloka
```

Ključna proširenja ES6 - let

- Za promenljive se uvodi ključna reč let, koja ima blok opseg važenja (block-scope)

```
{
  let year = 2020;
  …
}
//year ovde više nije dostupna
```

Ključna proširenja ES6 - const

- Podrška za konstante (nepromenjive varijable)

```
const PI = 3.141593
```

- Kao i varijable deklarisane sa let, const je block-scoped

- Napomena - ukoliko je varijabla objekat, samo referenca je u tom slučaju nepromenjiva, dok se sadržaj objekta ipak može menjati

Ključna proširenja ES6 - blok opseg važenja funkcija

- Poput varijabli i konstanti i funkcije sada imaju blok opseg važenja

```
{
    function foo () { return 1 }
    foo() === 1 // true
    {
        function foo () { return 2 }
        foo() === 2
    }
    foo() === 1
}
```

Ključna proširenja ES6 - Arrow functions

- Koncizna sintaksa pisanja funkcija
- Osim što se funkcija može pridružiti promenljivoj sada je moguće i skratiti zapis same definicije funkcije (može delovati malo nečitko na prvi pogled)
- pre

```
hello = function() {
  return "Hello World!";
}
```

- sada

```
hello = () => {
  return "Hello World!";
}
```

- a može i kraće
  (ako funkcija ima samo jedan izraz i ima return)

```
hello =()=> "Hello World!";
```

Ključna proširenja ES6 - Arrow functions

- Omogućava skraćivanje i pisanja callback funkcija bez potrebe pisanja function ključne reči

```
nums.forEach( v => {
    if (v % 5 === 0)
        fives.push(v)
})
```

- umesto

```
nums.forEach(function (v) {
    if (v % 5 === 0)
        fives.push(v);
});
```

Ključna proširenja ES6 - poboljšan rad sa parametrima funkcija

- Omogućava se eksplictino postavljanje podrazumevane vrednosti

```
function f (x, y = 7, z = 42) {
    return x + y + z
}
f(1) === 50
```

- umesto

```
function f (x, y, z) {
    if (y === undefined)
        y = 7;
    if (z === undefined)
        z = 42;
    return x + y + z;
};
f(1) === 50;
```

Ključna proširenja ES6 - „preostali parametri“

- „rest“ pattern
- Omogućava se agregacija preostalih parametara u jedan imenovani

```
function f (x, y, ...a) {
    return (x + y) * a.length
}
f(1, 2, "hello", true, 7) === 9 //true (1+2)*3
```

- umesto prethodnog

```
function f (x, y) {
    var a = Array.prototype.slice.call(arguments, 2);
    return (x + y) * a.length;
};
f(1, 2, "hello", true, 7) === 9;
```

Ključna proširenja ES6 - „spread operator“

- Omogućava se da se elementi neke kolekcije (a i stringa) „raspakuju“ u nekoj drugoj kolekciji, ili kao parametri funkcije

```
var params = [ "hello", true, 7 ]
var other = [ 1, 2, ...params ] // [ 1, 2, "hello", true, 7 ]

function f (x, y, ...a) {
    return (x + y) * a.length
}
f(1, 2, ...params) === 9

var str = "foo"
var chars = [ ...str ] // [ "f", "o", "o" ]
```

Ključna proširenja ES6 - destrukturisanje

- Destructuring assignement - omogućava da se elementi nekog niza ili objekta na intuitivan način i u skraćenom zapisu dodele varijablama

- umesto

```
var prvi  = niz[0];
var drugi = niz[1];
var treci = niz[2];
```

- dovoljno je sada

```
var [prvi, drugi, treci] = niz;
```

- moguće je i preskočiti nepotrebne elemente

```
var godine = ["2018", "2019", „2020“]
var [,,treci] = godine;
console.log(treci) //"2020"
```

Ključna proširenja ES6 - destrukturisanje

- Moguće je u „obuhvatiti“ sve preostale elemente koristeći šablon „preostalih“ elemenata („rest“ pattern)

```
var [head, ...tail] = [1, 2, 3, 4];
console.log(tail);  // [2, 3, 4]
```

Ključna proširenja ES6 - destrukturisanje objekata

- Omogućava da se varijablama dodele vrednosti različitih propertyja objekta
- U izrazu se navede naziv propertyja objekta koji se želi povezati sa varijablom, a zatim se navede i njen naziv

```
var robotA = { name: "Bender", year: 2020 };
var robotB = { name: "Flexo" , year: 2019 };

var { name: naziv1 } = robotA;  // var naziv1 = robotA.name
var { name: naziv2 } = robotB;

console.log(naziv1);
// "Bender"
console.log(naziv2);
// "Flexo"
```

Ključna proširenja ES6 - destrukturisanje objekata

- Omogućava i lepu „prečicu“ kada treba „raspakovati“ propertyje objekta u varijable sa istim nazivom (zgodno na primer kada treba „raspakovati“ request objekat).

- Dovoljno je u izrazu dodele navesti varijable koje imaju isti naziv kao traženi property-ji

```
var myObject = { prop1: "lorem", prop2: "ipsum" };

var { prop1, prop2 } = myObject;
console.log(prop1);
// "lorem"
console.log(prop2);
// "ipsum"
```

Ključna proširenja ES6 - destrukturisanje i podrazumevane vrednosti

- Kada se radi destrukturisanje moguće je odmah obezbediti i podrazumevanu vrednost za slučaj da traženi element kolekcije ili property objekta ne postoje

```
var [missing = true] = [];
console.log(missing);
// true

var { message: msg = "Something went wrong" } = {};
console.log(msg);
// "Something went wrong"

var { x = 3 } = {};
console.log(x);
// 3
```

Ključna proširenja ES6 - praktična primena destrukturisanja

- Definisanje parametara funkcije

- Često je moguće da ulazni parametar bude neki objekat (potencijalno sa mnogo propertyja), što je i zgodnije nego da korisnici pamte redosled parametara. Ako nama trebaju samo određeni, prosto destrukturiramo ulazni parametar izvlačeći samo one propertyje koji nama trebaju

```
function removeBreakpoint({ url, line, column }) {
  // ...
}
```

Ključna proširenja ES6 - praktična primena destrukturisanja

- Prilikom korišćenja konfiguracionih objekata kao parametara funkcija

```
jQuery.ajax = function (url, {
  async = true,
  beforeSend = noop,
  cache = true,
  complete = noop,
  crossDomain = false,
  global = true,
  // ... more config
}) {
  // ... do stuff
};
```

Ključna proširenja ES6 - praktična primena destrukturisanja

- Prilikom korišćenja ES6 iteracionih protokola

```
var map = new Map();
map.set(window, "the global");
map.set(document, "the document");

for (var [key, value] of map) {
  console.log(key + " is " + value);
}
```

- iteracija samo preko key ili samo preko value vrednosti

```
for (var [key] of map) {
  // ...
}

for (var [,value] of map) {
  // ...
}
```

Ključna proširenja ES6 - upotreba modula

- na nivou jezika je sada ugrađena podrška za upotrebu modula, što omogućava definisanje komponenti u sopstvenim fajlovima.
- Ostvaruje se pomoću export i import ključnih reči

```
// lib/math.js
export function sum(x, y) {
  return x + y;
}
export var pi = 3.141593;
```

```
// app.js
import * as math from "lib/math";
alert("2π = " + math.sum(math.pi, math.pi));
```

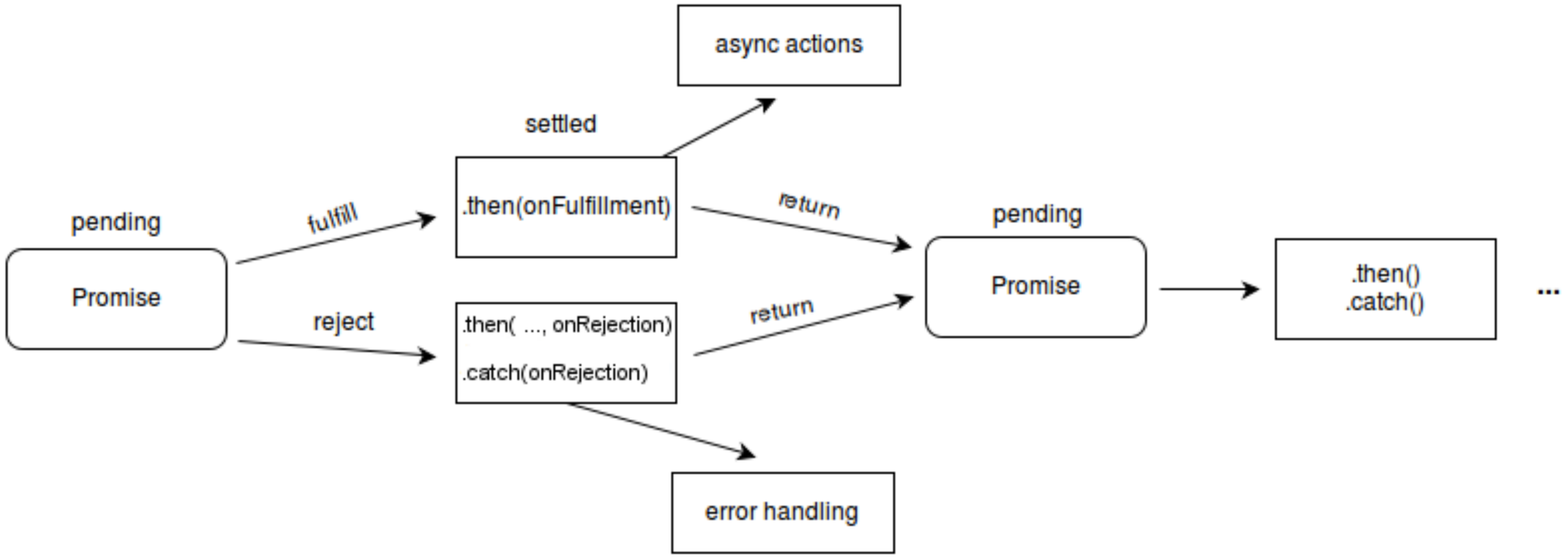
Ključna proširenja ES6 - Promise

- Promise je objekat namenjen za podršku radu sa asinhronim pozivima.
- Sadrži kako produkcioni kod (koji proizvodi rezultat), tako i poziv za kod koji konzumira nastali rezultat
- Predstavljaju vrednosti koje mogu postati dostupne u budućnosti.
- Promise objekat može da bude u tri stanja:
  - pending
  - fulfilled
  - rejected

Ključna proširenja ES6 - Promise

- Metode then i catch se koriste da se u njima specificira šta se radi kada Promise dođe u određeno stanje

Ključna proširenja ES6 - Promise



Ključna proširenja ES6 - Promise

```
myPromise.then(
   function(value) { /* code if successful */ },
   function(error) { /* code if some error */ }
);
```