JavaScript podrska u radu sa greskama

Svaki od pregledaca ima svoj podrazumevani naci reagovanja na greske, npr. Firefox i Chrome upisuju greske u log datoteku, dok recimo Internet Explorer i Opera generisu obavestenja za korisnika.

Zbog udobnosti rada korisnika potrebni su nam mehanizmi koji bi omogucili rukovanje greskama (pracenje njihovog pojavljivanja i reagovanje na njih) kao vid prevencije reagovanja pregledaca.

Greske se mogu obradjivati u okviru **try-catch** bloka: ovakve greske pregledaci smatraju primecenim pa ne reaguju na njih.

```
try{
        …
}catch(error){
        …
}
```

Tipovi gresaka koji mogu da se jave:

Error – osnovni tip greske (svi drugi tipovi ga nasledjuju)
ima svojstvo *message* sa opisom greske i svojstvo *name* kojim se odredjuje tip greske pregledaci interno obogacuju ovaj objekat pa npr. Firefox ima i svojstva *fileName* sa imenom datoteke u kojoj se javila greska, *lineNumber* koji sadrzi broj linije greske i *stack* koji sadrzi stack trace.

EvalError – greska koja se generise u radu sa *eval()* funkcijom
eval() - funkcija izvrsava JavaScript kod zapisan u vidu niske i prosledjen kao argument
npr. eval("var a, b,c; a=5; b=10; c=a+b");

eval() funkciju treba koristiti uz oprez ukoliko postoji mogucnost da njen argument postane maliciozni kod koji moze da ugrozi podatke i aplikaciju
uz sve to, *eval()* se nesto sporije izvrsava jer eskplicitno poziva JS interpreter

RangeError – greska koja se generise kada vrednost koja se navodi nije u skupu ili opsegu dozvoljenih vrednosti npr. a=new Array(-20)

ReferenceError – javlja se u slucajevim kada se referise na nepostojecu promenljivu
npr. *a=x;* //dok *x* ne postoji

SyntaxError – obicno se javlja kada se prosledi sintaksno neispravan kod funkciji *eval();* u svim ostalim slucajevima sintaksne greske automatski prekidaju izvrsavanje JavaScript koda

TypeError – tip greske koji se najcesce javlja i to kada je promenljiva neodgovarajuceg tipa ili kada se pokusava sa pristupom metodi koja ne postoji.

URIError – greska koja se javlja kao posledica koriscenja funkcija encodeURI() i decodeURI() na URI argumentima koji su pogresnog formata

encodeURI() - funkcija koja vrsi samo kodiranje URI-ja
kodiraju se svi specijalni karakteri (%HH) osim
rezervisanih karaktera: ; , / ? : @ & = + $ #
slova
cifara
i karaktera - _ . ! ~ * ' ( )

decodeURI() - invernza funkcija funkcije encodeURI

encodeURIComponent() - funkcija koja vrsi kodiranje nesto sireg skupa karaktera
sve osim slova, cifara i - _ . ! ~ * ' ( )
pa se onemogucava dodavanje novih parova URI-jima
ime=vrednost u slucaju
application/x-www-form-urlencoded formata

decodeURIComponent() - inverzna funkcija funkcije encodeURIComponent()

greske se generisu sa ***throw***

```
function test(x){

	if(typeof x != “number”)
		throw new TypeError(...);

	if(x < conditions.MIN_VALUE || x>conditions.MAX_VALUE )
		throw new RangeError(...);

	return true;
}
```

sa ***error instanceof*** .... se moze utvrditi kojeg je tipa greska koja se pojavila i u skladu sa tim preuzeti odgovarajuci skup akcija

```
try{
        var x=”abc”;
        test(x);
}catch(error){
        if( error instanceof TypeError){
                ….....
        }
        else
                if(error instanceof RangeError){
                        ….............
                }
                else{
                        console.log(error.message);
                }
}
```

Mogu se kreirati i vlastiti objekti koji nasledjuju klasu Error

```
function myError(message){
        this.message=message;
        this.name=”myError”;
}

myError.prototype=new Error();
```

u kodu: … throw new myError(“my message”);

- Upisivanje u centralizovani log sistem (na strani servera): svaki put kada se greska javi moze se generisati AJAX poziv koji sadrzi tip greske i opis greske; tako se lakse moze pratiti ponasanje aplikacije na strani klijenta

npr.

```
        //tzv. image pings tj. slanje zahteva preko img objekta
        function logerror(type, message){
                var img=new Image();
                img.src=”log.php?type=”+encodeURIComponent(type)+”&message=” +
                                encodeURIComponent(message);
        }

        u kodu
        ...
        catch(error){
                if(error instanceof SyntaxError){
```

```
                logerror("syntax", "Description: "+error.message);
        }
        …
}
```

Image pings je jednosmerni vid komunikacije izmedju klijenta i servera koji zaobilazi *the same origin policy* (moze se komunicirati izmedju raznorodnih domena).

- Cesto se koriste za pracenje korisnickih klikova
- mana im je sto postoji mogucnost slanja samo GET zahteva i sto ne postoji mogucnost obrade serverskog tekstualnog odgovora: na raspolaganju su samo dogadjaji *onload* i *onerror* kojima se moze ispratiti kada pristigne odgovor od servera – to je dalje obicno ili slika ili kod 204

  npr. img.onload=img.onerror=function(){ …obavestiti korisnika o uspesnosti... }

STRICT MODE

- pojavio se sa ECMA5 standardom
- podrzavaju ga svi aktuelni pregledaci
- omogucava stroziju kontrolu gresaka bilo globalno bilo na nivou pojedinacnih funkcija
- u striktni mod se ulazi navodjenjem "use strict"; niske
- na globalnom nivou

```
"use strict";
function ….
function ….
```

(ukljucivanjem jedne datoteke sa globalnim *strict* modom rezultira *strict* modom na nivou svih datoteka)

- lokalno, na nivou funkcije

```
function test(....){
        "use strict";
        …..
        …..
}
```

- ideja je spreciti sve one greske preko kojih interpreter moze da predje (silently falls)

npr.

deklaracija u globalnom opsegu

message="....";

ce rezultirati sa ReferenceError je se tezi sprecavanju slucajno kreiranih globalnih promenljivih

npr.

Pridruzivanje vrednosti read-only svojstvu
rezultat: TypeError

npr.

Koriscenjem delete nad objektom kod koga je configurable svojstvo postavljeno na false
rezultat: TypeError

npr.

Prosirivanje (dodavanje novog svojstva) objektu za koji je extensible postavljeno na false
rezultat: TypeError

- eval() funkciji se zabranjuje uvodjenje novih promenljivih i funkcija
  npr.
  eval("var x=10;");
  alert(x);
  generise SyntaxError

- ogranicenja na imenovanje promenljivih
  npr.
  eval, arguments, … nisu dozvoljena imena
  rezultat: je SyntaxError