



Distance Learning System

Advanced JavaScript

Instanciranje funkcija i varijabli

Instanciranje funkcija i varijabli

- *Instanciranje top-level varijabli*
- *Varijable funkcija*
- *Blokovi nemaju oblast važenja (scope)*

Instanciranje top-level varijabli

- Mehanizam po kom funkcionišu varijable i funkcije u JavaScript-u je drugačiji od gotovo svih drugih programskih jezika.
- U JavaScriptu sve globalne varijable i funkcije su u stvari svojstva specijalnog objekta koji se zaniva LexicalEnvironment što je u slučaju upotrebe JavaScripta u WEB razvoju zapravo prozor pretraživača. Ovaj objekat se još naziva i **globalni objekat**.

Instanciranje top-level varijabli

- U trenutku kada JavaScript započne izvršavanje dešava se faza koji prethodi svim ostalim fazama procesa (pre-processing stage) koja se naziva **variables instantiation**.
- Prvi korak ove faze jeste skeniranje koda i smeštanje funkcija u window objekat. Primer:

```
var a = 5
```

```
function f(arg) { alert('f:'+arg) }
```

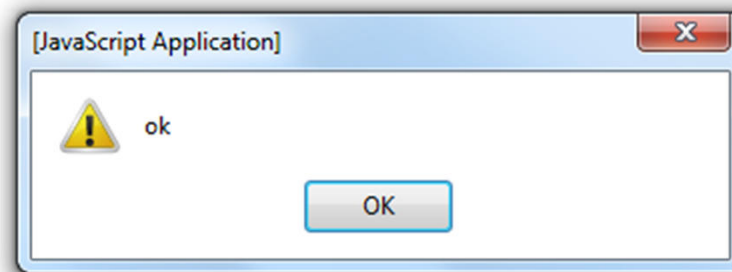
```
var g = function(arg) { alert('g:'+arg) }
```

- Kod ovog primera browser najpre pronalazi funkciju f, kreira funkciju i smešta je u window.f

Instanciranje top-level varijabli

- Zbog ponašanja objašnjenog prethodnim slajdom funkcije je moguće pozivati i pre njihovog deklarisanja, što nije slučaj u nekim drugim programskim jezicima. Primer:

f()
function f() { alert('ok') }



Instanciranje top-level varijabli

- U ovoj fazi (prva faza) interpreter skenira **var** deklaracije i kreira svostva window objekta.
- Napomena: Dodeljivanje vrednosti se ne vrši u ovoj fazi. Sve varijable dobijaju početnu vrednost *undefined*.
- U prethodnom primeru:

```
var a = 5;
```

```
function f(arg) { alert('f:'+arg) }
```

```
var g ← function(arg) { alert('g:'+arg) }
```

bi u ovoj fazi window objekat izgledao ovako:

```
window = { f: function, a: undefined, g: undefined }
```

Instanciranje top-level varijabli

- Napomena:

```
var a = 5;  
function f(arg) { alert('f:'+arg) }  
var g = function(arg) { alert('g:'+arg) }
```

Uočavamo da u ovom primeru promenljiva **g** ima vrednost funkcije, ali ne smemo dozvoliti da nas ovo prevari jer interpreter ne obraća pažnju na sadržaj varijable. Iako je u pitanju funkcija inicijalno će promenljiva **g** dobiti vrednost *undefined*.

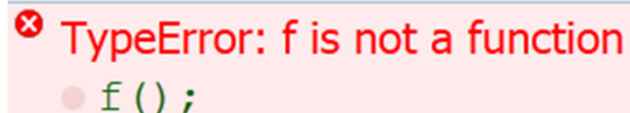
```
window = { f: function, a: undefined, g: undefined }
```

Instanciranje top-level varijabli

- Propratni neželjeni efekat ovakve logike rada programskog jezika jeste nemogućnost postojanja varijable i funkcije sa istim imenom (identifikatorom).

- Primer:

```
function f(arg) { alert('f:'+arg) }  
var f = 10;  
alert(f);  
f();
```

A screenshot of a JavaScript error message in a light pink box. It features a red 'x' icon, the text 'TypeError: f is not a function' in red, and a line of code 'f ();' in green with a red dot indicating the error location.

✖ TypeError: f is not a function
● f ();

Instanciranje top-level varijabli

- Pogledajmo na primeru različite slučajeve kojima se prikazuje logika rada u počenoj fazi:

```
alert("a" in window) // true, zato što window.a postoji  
alert(a) // undefined, zato što se dodeljivanje vrednosti dešava kasnije  
alert(f) // function, zato što je ovo deklaracija funkcije  
alert(g) // undefined, zato što se dodeljivanje vrednosti dešava kasnije  
bez obzira na sadržaj promenljive
```

```
var a = 5  
function f() { /*...*/ }  
var g = function() { /*...*/ }
```

Instanciranje top-level varijabli

- U sledećem primeru se alert funkcije pozivaju nakon dodeljivanja vrednosti:

```
var a = 5  
var g = function() { /*...*/ }
```

```
alert(a) // 5  
alert(g) // function
```

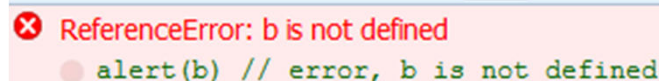
Instanciranje top-level varijabli

- Ukoliko varijabla nije deklarirana uz pomoć naredbe **var** ona naravno neće biti kreirana u fazi inicijalizacije, odnosno interpreter je neće videti. Primer:

```
alert("b" in window) // false, jer ne postoji window.b
```

```
alert(b) // error, b is not defined
```

```
b = 5
```

A screenshot of a JavaScript error message in a console. It features a red 'x' icon in a circle, followed by the text 'ReferenceError: b is not defined' in red. Below this, a light blue line of code is shown: 'alert(b) // error, b is not defined'.

Instanciranje top-level varijabli

- Za razliku od primera sa prethodnog slajda ovde je promenljiva **b** upotrebljena nakon deklarisanja i zato nema grešaka u kodu:

```
b = 5
```

```
alert("b" in window) // true, jer postoji window.b = 5
```

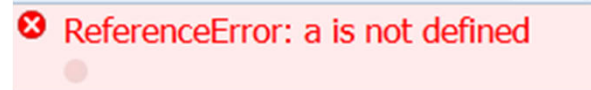
```
alert(b);
```

Instanciranje top-level varijabli

- Pitanje 2:

Šta će se dogoditi nakon sledećeg koda?

```
if ("a" in window) {  
    a = 1;  
}  
alert(a);
```

A red error message box with a white 'x' icon and the text "ReferenceError: a is not defined".

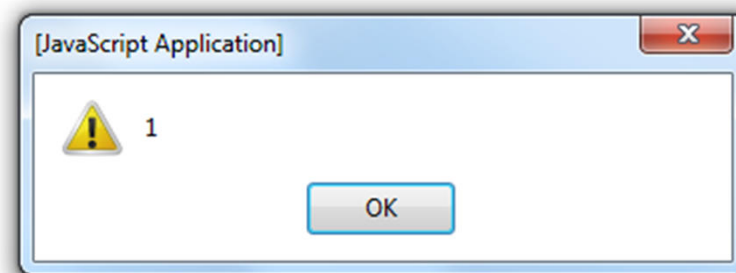
ReferenceError: a is not defined

Instanciranje top-level varijabli

- Pitanje 1:

Šta će se dogoditi nakon sledećeg koda?

```
if ("a" in window) {  
    var a = 1;  
}  
alert(a);
```



Varijable funkcija

- Kada se funkcija pokrene, pri svakom pozivu, kreira se novi LexicalEnvironment i ispunjava se argumentima, promenljivama i unutrašnjim funkcijskim deklaracijama.
- Za razliku od window objekta, LexicalEnvironment funkcije nije otvoren za pristup spolja (*direct access*) već mu se može pristupiti samo iz tela funkcije.

Varijable funkcija

- Dok se interpreter priprema da izvrši funkciju, dakle pre izvršenja prve linije koda iz tela funkcije, kreira se prazan LexicalEnvironment i ispunjava se argumentima, lokalnim varijablama i ugrađenim funkcijama. Primer:

```
function sayHi(name) {  
  // LexicalEnvironment = { name: 'John', text: undefined }  
  var text = "Hi, " + name;  
  alert(text);  
}  
sayHi('John')
```

Uobičajeno je da u ovoj fazi argumenti poseduju vrednost dok ih varijable još uvek nemaju.

Varijable funkcija

- Nakon inicijalnog dela koji je opisan prethodnim slajdom događa se sledeći korak u kojem se vrši eventualna dodela vrednosti lokalnim varijablama:

```
function sayHi(name) {  
  // LexicalEnvironment = { name: 'John', text: undefined }  
  var text = "Hi, " + name;  
  // LexicalEnvironment = { name: 'John', text: 'Hi, John'}  
  alert(text);  
}  
sayHi('John');
```