► Hyrje në JavaScript

"JavaScript shton ndëraktivitetin në website" .

JavaScript punon së bashku me **HTML** dhe **CSS** për krijimin e websiteve ndëraktive dhe tërheqëse.

*Para se të filloni të mësoni JavaScript rekomandohet të keni njohuri bazë në **HTML** dhe **CSS**.*

► Përdorimi i JavaScript

Në faqet web javascript punon brenda **web browser-it**.

Ja disa shembuj se për çfarë mund të përdoret JavaScript:

- **Për të shfaqur informacion në bazë të datës së ditës**
- **Dedektimin e browser-it të përdoruesit**
- **Validimin e të dhënave të formave**
- **Krijimin e** cookie-ve
- **Ndryshimin dinamik të përmbajtjeve të faqeve web**
- dhe shumë më tepër!

►Një gjuhë Client Side

Script-et në website ekzekutohen si **në anën klient** dhe në **anën server**.

Ana client-side e një website-i i referohet browserit që do ta shoh atë. Ana server-side e një website-i i referohet serverit që do ta hostojë atë. PHP, Ruby on Rails, ASP.NET janë gjuhët më popullore server-side. Ato quhen gjuhët server-side sepse ekzekutohen në serverin që hoston website-in dhe jo në kompjuterin e përdoruesit.

*JavaScript është gjuhë skriptive **Client Side**.*

JavaScript **ekzekutohet në kompjuterin tuaj** pasi në browser ngarkohet faqa web që përmban atë.

► Çfarë na duhet pët të punuar?

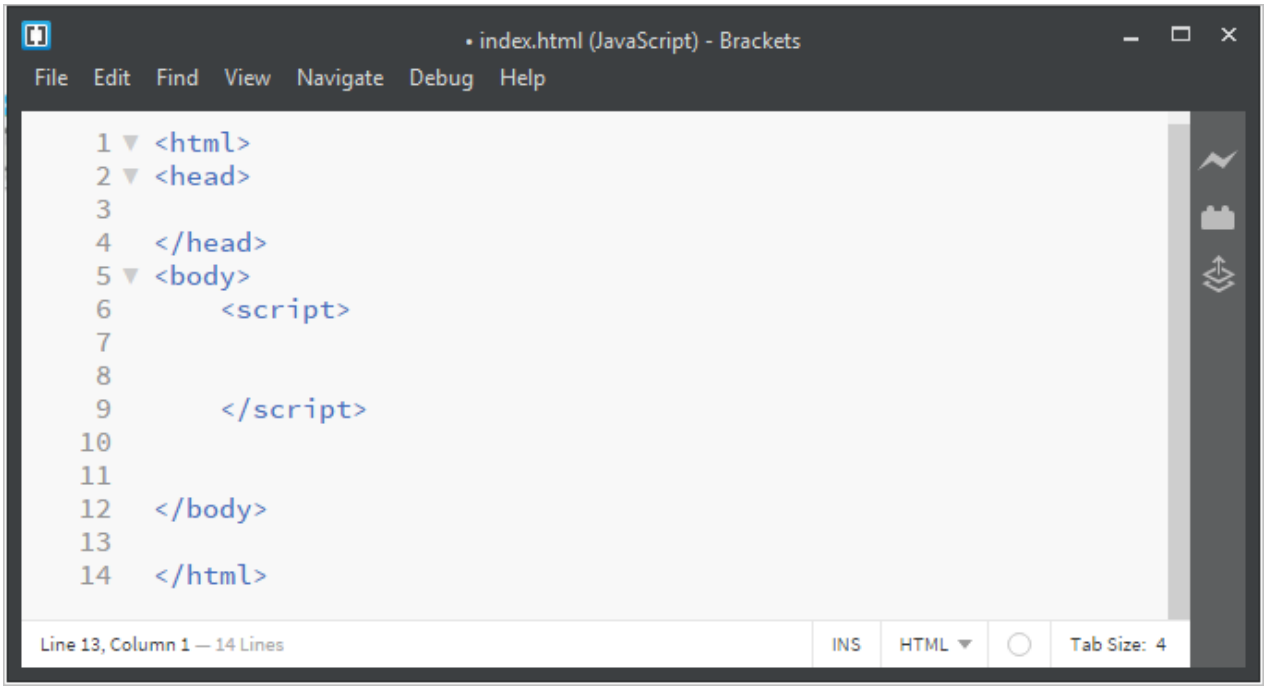
Një nga avantazhet e JavaScript është se nuk kërkohen mjete speciale për të punuar. Gjithçka ju duhet është një **text editor**.

Gjatë shpjegimit këtu është përdorur editori open source **bracket.**

► Fillo shkruaj JavaScript

JavaScript mund të shkruhet brenda një dokumenti **HTML**.
Në HTML, kodi JavaScript duhet të shkruhet brenda tageve **<script> dhe </script>** :
Këto tageve mund të shkruhen brenda tageve body dhe head të **HTML-së.**

Në shembullin më poshtë është shkruajtur brenda tageve **<body> .**

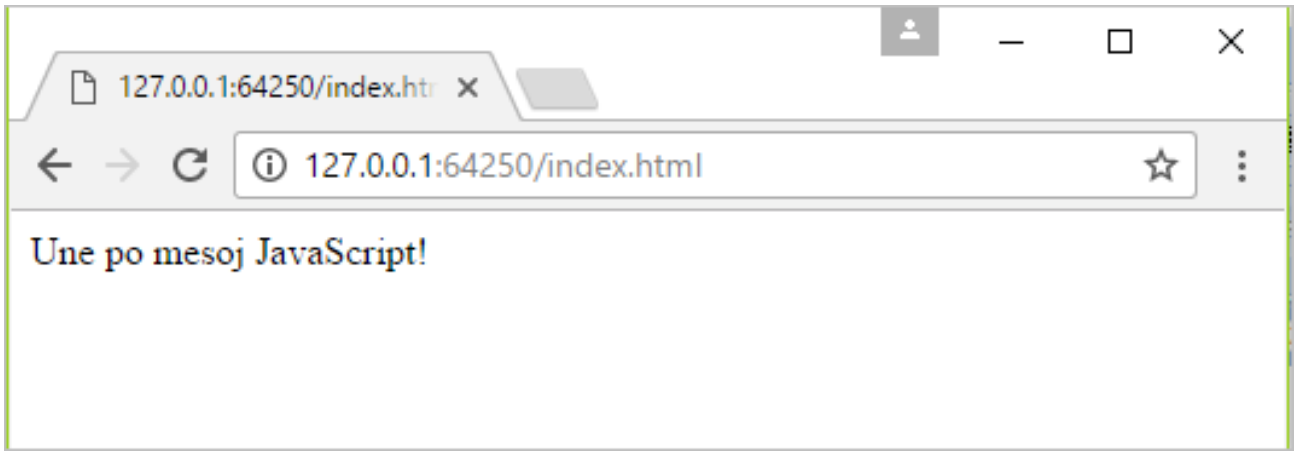


Le të përdorim **JavaScript** për të printuar në browser **"Unë po mësoj JavaScript!"**

```
<html>
<head> </head>
<body>
<script>
document.write("Une po mesoj JavaScript!" );
</script>
</body>
</html>
```

Funksioni document.write() shkruan një string në dokumentin HTML.

Kodi i mësipërm do të na japë rezultatin e mëposhtëm:



► Formatimi i Tekstit

Si në HTML, ne mund të përdorim taget **HTML** për formatimin e tekstit në **JavaScript.**

Për shembull:

```
<html>
<head> </head>
<body>
<script>
document.write("<h1>Une po mesoj JavaScript!</h1>");
</script>
</body>
</html>
```

Shtimi i javaScript-it në një faqe web:

► JavaScript në <head>

Një dokument HTML mund të ketë disa scripte.
Më poshtë javascript është vendosur brenda tageve head.

```
<html>
<head>
<script>
</script>
</head>
<body>
</body>
</html>
```

► JavaScript në <body>

Scriptet JavaScript shkruhen brenda tageve **<body> .**

```
<html>
<head> </head>
<body>
<script>
</script>
</body>
</html>
```

Rekomandohet vendosja e tageve script në fund të tageve <body> .

Kjo përmirëson ngarkimin e faqeve web, sepse shfaqja e tageve HTML nuk do të bllokohet nga ngarkimi i scripteve.

▶Tagu <script>

Tagu **<script>** merr dy atribute, **language** dhe **type,** të cilat specifikojnë tipin e script-it dhe gjuhën**:**

```
<script language="javascript" type="text/javascript"> </script>
```

Në shembullin e mëposhtëm , ne krijojmë një dritare **alert** brenda scripteve duke përdorur funksionin **alert().**

```
<html>
<head>
<title></title>
<script type="text/javascript">
alert("Kjo është një dritare alert!");
</script>
</head>
<body>
</body>
</html>
```

Rezultati:

Tipi atributit:

<script type="text/javascript"> nuk përdoret më , përderisa JavaScript është gjuha primare skriptive për HTML.

► JavaScript i jashtëm

Skriptet mund të shkruhen gjithashtu edhe në skedar të jashtëm.
Skriptet e jashtme janë **më tepër produktive** në rastin kur lidhen me shumë web faqe .
Skedarët JavaScript kanë prapashtesën js.

Për të përdorur një skript të jashtëm , vendosim emrin e skedarit të skriptit brenda atributeve src (source) të tagut <script>.

Shembull:

```
<html>
<head>
<title> </title>
<script src="javascript.js"></script>
</head>
<body>
</body>
</html>
```

! Avantazhet e përdorimit të skedarëve të jashtëm :

- Ndan HTML dhe kodin script.
- Lexim dhe mirëmbajtje më të thjeshtë për skedarët HTML dhe JavaScript
- Ngarkim më i shpejtë i faqeve web.

► Komentet në JavaScript

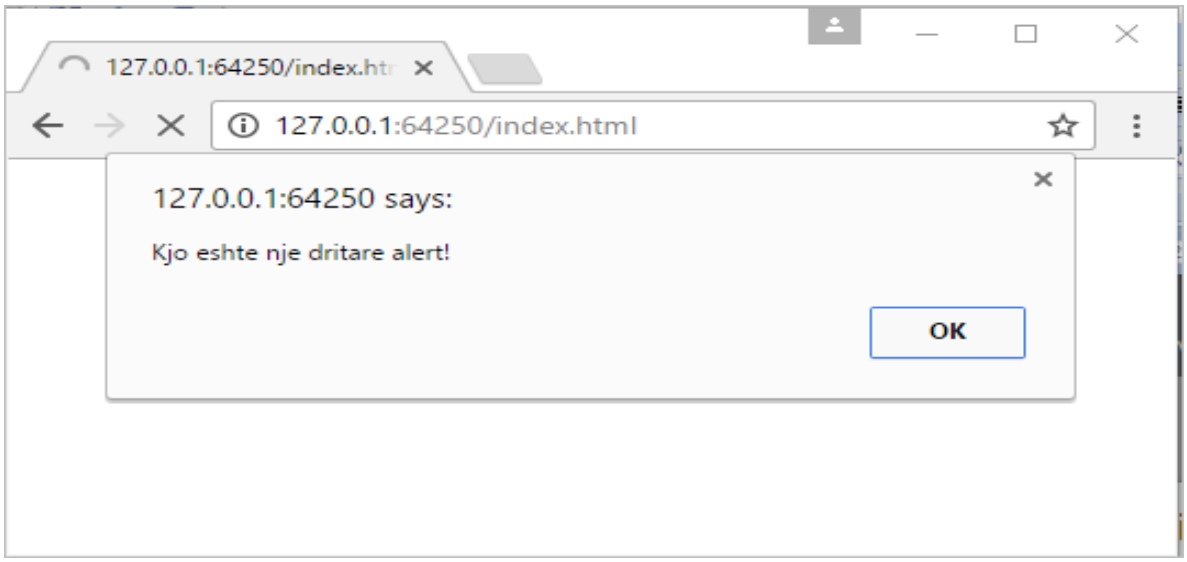
Për shpjegimin e pjesëve të caktuara të kodit përdoren komentet.
Ato shkruhen pas //, ose midis /* dhe */.

Komentet injorohen nga browser-at dhe nuk ekzekutohen.
Komentet e vetëm një rreshti shkruhen pas //

```
<script>
// Ky është koment i një rreshti

alert("Kjo eshte ne dritare alert!");
</script>
```

Rezulati:



► Komentet e shumë rreshtave

Çdo gjë ju shkruani midis /*dhe */ do të konsiderohet koment shumë rreshtësh.

Ja një shembull.

```
<script>
/* Ky kod

krijon nje

dritare  alert

 */
alert("Kjo eshte nje dritare alert!");
</script>
```

► Konceptet bazë

Variablat

Variablat janë mbajtës të vlerave të të dhënave. Vlera e një programi do të ndryshojë gjatë programit. Për deklarimin e variablave përdoret fjala kyçe **var** : **var** x = 100;
Në shembullin e mësipërm variablës **x** i jepet vlera **100**.

JavaScript është case sensitive.

Për shembull, variabla ***emri*** **dhe** ***Emri*,** janë dy variabla të ndryshme.

Shenja e barazimit

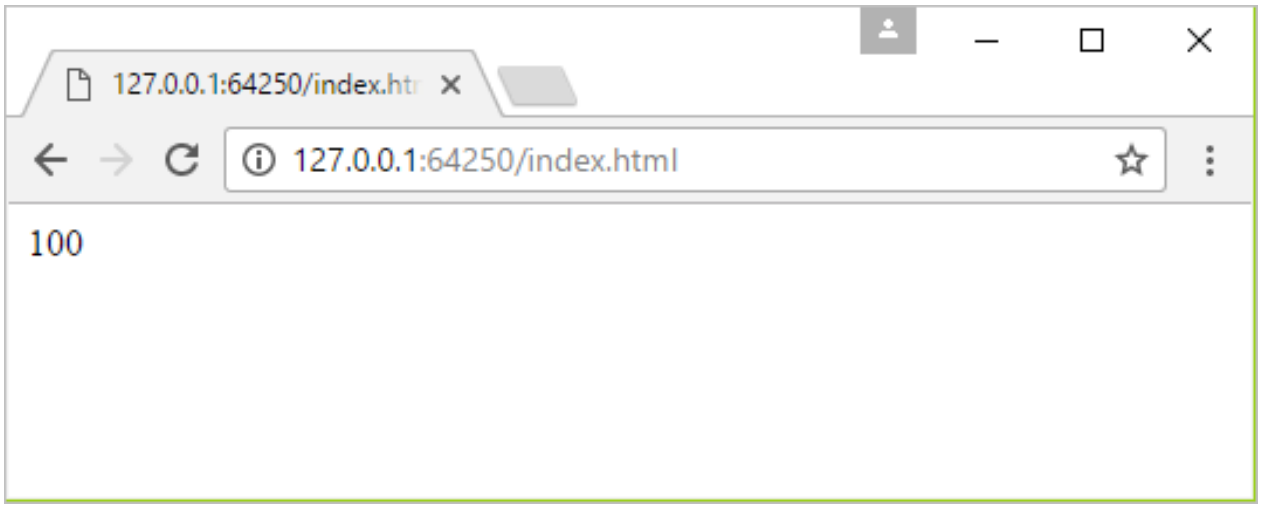
Në JavaScript, shenja e (=) quhet operatori "**dhënies së vlerave**" dhe jo barazimit .
Për shembull, **x = y** ku x-it i jepet vlera e **y** .
Një variabël mund të deklarohet pa dhënie vlere.

Përdorimi i Variableve

Dhënia e vlerës 100 variablës x.

```
var x = 100;
document.write(x);
```

Rezultati:



Çdo instruksion JavaScript ndahet me pikëpresje.

Emërtimi i variablave

Emrat e variablave JavaScript janë case-sensitive.
Për shembull:

```
var x = 100;
document.write(X);
```

Ky kod nuk do të japë asnjë output perderisa x dhe X janë y variabla të ndryshme.

Rregullat e emërtimit të variablave në JavaScript:

- Karakteri i parë **duhet të jete** një shkronjë, një underscore (_), ose një shenjë dollari ($).

Karakteret e tjera mund të jenë shkronja, numra, underscore, ose shenja dollari.

- Numrat nuk lejohen si karaktere fillestare.
- Emrat e variablave **nuk** mund të përfshijnë operator matematik ose logjikë.
- Nuk duhet të përmbajnë hapësira.
- Nuk duhet të përdoren karaktere speciale, si *num#ber, num%,* etj.
- Dhe nuk lejohet emërtimi i variablave me një nga fjalët e mëposhtme të rezervuara.

Fjalët e rezervuara në Java Script

abstract	else	instanceof	switch
boolean	enum	int	synchronized
break	export	interface	this
byte	extends	long	throw
case	false	native	throws
catch	final	new	transient
char	finally	null	true
class	float	package	try
const	for	private	typeof
continue	function	protected	var
debugger	goto	public	void
default	if	return	volatile
delete	implements	short	while
do	import	static	with
double	in	super	

► Tipet e të dhënave

Termi **tipe të dhënash i referohet** tipit të vlerës së një variable.

Variablat JavaScript mund të mbajnë disa tipe të dhënash,
si **numbrat**, **stringjet**, **matricat**, dhe shumë më tepër.

Ndryshe nga gjuhët e tjera të programimit, JavaScript nuk bën ndarje midis llojeve të ndryshme të numrave si : integers, short, long, floating-point, etj.
Numrat në JavaScript mund të shkruhen me ose pa presje.

Stringjet

Stringjet në JavaScript shkruhen për manipulimin e tekstit.
Stringjet janë fjalë që shkruhen me thonjëza teke ose dyshe.

```
var titulli= 'JavaScript';
var fjali = "Une mesoj JavaScript";
```

Boolean

Vlerat Boolean-e në JavaScript, mund të marrin dy vlera **true** ose **false**.
Këto lloj vlerash përdoren kur mjafton që variablat të kenë dy vlera, si Po/Jo, On/Off, True/False.

► Operatorët Aritmetikë

Operatorët aritmetik performojnë operacione aritmetike me variablat.

Për shembull:

```
var x = 100 + 15;
document.write(x);

// Output 115
```

Shumës mund t'i shtohen sa variabla të jetë e nevojshme.

```
var x = 100;
var y = x + 1+ 4 + 200;
document.write(y);

//Output 305
```

Shumëzimi

Operatori i shumëzimit (*):

```
var x = 100 * 5;
document.write(x);

// Output 500
```

Pjestimi:

Operatori / veprimin aritmetik të pjestimit:

```
var x = 200 / 5;
document.write(x);

// Outputi 4
```

Moduli

Operatori modulit (%)

```
var numri= 26 % 6;

//Output do te jete 2
```

► Inkrementimi dhe dekrementimi

Inkrementimi ++

Operatori i inkrementimit rrit vlerën e operatorit me 1. Nese vendoset pas operandit, kthen vlerën origjinale më pas inkrementon operandin

Dekrementimi --

Operatori i dekrementimit zbret vlerën e operatorit me 1. Nese vendoset pas operandit, kthen vlerën origjinale më pas dekrementon operandin

---

► Operatorët e dhënies së vlerave

Operatori	Shembull	Njëvlershmëria
=	x = y	x = y
+=	x += y	x = x + y
-=	x -= y	x = x -y
*=	x *= y	x= x * y
/=	x /= y	x = x / y
%=	x %= y	x = x % y

► Operatorët e krahasimit

Operatorët e krahasimit bëjnë krahasimin midis dy operandeve dhe kthejnë vlerën true ose false**.**

Operatori	Përshkrimi	Shembull
==	E barabartë	10==100 false
===	Identike (i njëjti tip)	10===100 false
!=	Jo e barabartë	5!=100 true
!==	Jo Identike	10!==10 false
>	Më e madhe	10 > 100 false
>=	Më e madhe ose e barabartë	10>=100 false
<	Më e vogël	10 < 100 true
<=	Më e vogël ose e barabartë	10 <= 100 true

► Operatorët logjik ose boolean

Operatorët logjik, gjithashtu të njohur si operatorët **Boolean** , vlerësojnë shprehjen dhe kthejnë true ose false.
Tabela e mëposhtme shpjegon operatorët logjik (**AND**, **OR**, **NOT**).

	Operatorët Logjikë
&&	Kthen true nëse të dy operandët janë të vërtet
||	Kthen true nëse të paktën një nga operandët është i vërtet
!	Kthen true nëse operarandi është false dhe false kur operandi është true

► Operatorët e stringjeve

Operatori më i përdorshëm për stringjet është bashkimi, me anë të shenjës së +. Bashkimi mund të përdoret për ndërtimin e stringjeve duke bashkuar bashkë shumë stringje.

```
var string1 = "Une po mesoj ";
var string2 = "Javascript.";
document.write(string1 + string2);
```

Shembulli i mësipërm i deklaron dhe i inacializon variablat më pas i inacializon ato.

► Kushtëzimet

Kushti If

Shpesh kur shkruajmë kod, ne duam të performojmë veprime të ndryshme bazuar në kushte të ndryshme .

Ju mund ta bëni këtë duke përdorur thëniet kushtëzuese në kodin tuaj.

Përdorni **if** për specifikimin e një blloku kodi që do të ekzekutohet nëse një kusht specific do të jetë i vërtetë.

```
if (kushti) {
instruksione
}
```

Instruksionet do të ekzekutohen vetëm nëse kushti do të jetë i vërtetë.

Shembull:

```
var numri1 = 8;
var numri2 = 100;
if (numri1 < numri2) {
```

```
alert("JavaScript është e thjeshtë për t'u mësuar.");
}
```

Përdorimi i **else**

```
var numri1 = 5;
var numri2 = 100;
if (numri1 > numri2) {
alert("Ky është kushti parë");
}
else {
alert("Ky është kushti dytë");
}
```

Shembulli i mësipërm thotë:
- **If numri***1* është më i madh se numri*2*, alert "Ky është kushti parë";
- **Else**, alert "Ky është kushti dytë".

Browser-i do të printojë kushtin e dytë .

else if

Instruksioni else if përdoret për specifikimin e një kushti të ri nëse kushti i parë është i gabuar.

Shembull:

```
var kursi = 1;
if (kursi == 1) {
document.write("<h1>HTML Shqip</h1>");
} else if (course == 2) {
document.write("<h1>CSS Shqip</h1>");
} else {
document.write("<h1>JavaScript Shqip</h1>");
}
```

Pra, sipas kodit të mësipërm:

if (nese) kursi është i barabartë me 1, output "HTML Tutorial";
- **else, if** (nese) kursi është i barabartë me 2, output "CSS Tutorial";
- Nëse asnjë nga instruksionet e mësipërme nuk është e vërtetë, output "JavaScript Tutorial";
Mund të shkruhen sa instruksione **else if** të jetë e nevojshme.

Switch

Në rast se janë për t'u testuar shumë kushte, shkrimi i instruksionit **if else** për çdo kusht, nuk është zgjidhja e duhur. Instruksioni **switch** përdoret për performimin e veprimeve të ndryshme bazuar në kushte të ndryshme.

Sintaksa:

```
switch (shprehje) {
case n1:
instruksione
break;
case n2:
instruksione
break;
default:
instruksione
}
```

Vlera e shprehjes krahasohet me çdo bllok . Nëse ka përputhje, do të ekzekutohet blloku i kodit i cili është për atë rast.
I njëjti rezultat mund të arrihet edhe me shumë blloqe if..else por përdorimi i switch është më shumë efektivë.

Instruksioni switch

Marrim në studim shembullin e mëposhtëm:

```
var muaji = 2;
switch (muaji) {
case 1:
document.write("Janar");
```

```
break;
case 2:
document.write("Shkurt");
break;
case 3:
document.write("Mars");
break;
default:
document.write("Një muaj tjetër");
}

// Output-i "Shkurt"
```

Mund të përdoren sa instruksione case të jetë e nevojshme.

Fjala kyçe break

Kur JavaScript arrin një **break**, kalon jashtë bllokut switch.
Kjo do të ndalojë ekzekutimin e shumë blloqeve të kodit brenda switch për secilin case.
Zakonisht, **break** duhet të përdoret pas çdo instruksioni case.

Fjala kyçe default

Fjala kyçe default specifikon se çfarë kodi do të ekzekutohet nëse nuk përputhet asnjë nga instruksionet case.

```
var ngjyra ="verdhë";
switch(ngjyra) {
case "blu":
document.write("Kjo është ngjyrë blu.");
break;
case "kuqe":
document.write("Kjo është ngjyrë e kuqe.");
break;
case "gjelbërt":
```

```
document.write("Kjo është ngjyrë e gjelbërt.");
break;
case "portokalli":
document.write("Kjo është ngjyrë portokalli.");
break;
default:
document.write("Ngjyra nuk u gjet.");
}

//Output-i " Ngjyra nuk u gjet."
```

► Loops- Ciklet

Ciklet mund të ekzekutojnë një bllok kodi disa herë. JavaScript ka tre lloje ciklesh: **for**, **while**, dhe **do while**.

Cikli for

Sintaksa:

```
for (instruksioni 1; instruksioni 2; instruksioni 3) {
blloku i kodit që do të ekzekutohet
}
```

Instruksioni 1 ekzekutohet para se të fillojë cikli.
Instruksioni 2 përcakton kushtin e ekzekutimit të ciklit
Instruksioni 3 ekzekutohet gjithmonë pasi cikli të jetë ekzekutuar njëherë.
Pra, cikli for ka tre instruksione ose parametra.

Marrim në studim shembullin e mëposhtëm :

Krijimi i një cikli for i cili printon të gjithë numrat nga 1 deri në 10.

```
for (i=1; i<=10; i++) {
document.write(i + "<br />");
}
```

Në këtë shembull, **Instruksioni 1** i jep vlerë variablës para se të fillojë ekzekutimi i ciklit (var i = 1).
Instruksioni 2 përcakton kushtin e ekzekutimit të ciklit.
Instruksioni 3 rrit vlerën e variablës sa herë që ekzekutohet blloku i kodit.

Instruksioni 1 është opsional, për shembull:

```
var i = 1;
for (; i<=10; i++) {
document.write(i + "<br />");
}
```

Gjithashtu, mund të inicializohen më shumë se një vlerë duke përdorur presje për t'i ndarë.

Cikli While

Cikli **while** përsërit një blok kodi, për sa kohë një kusht i caktuar është i vërtet .

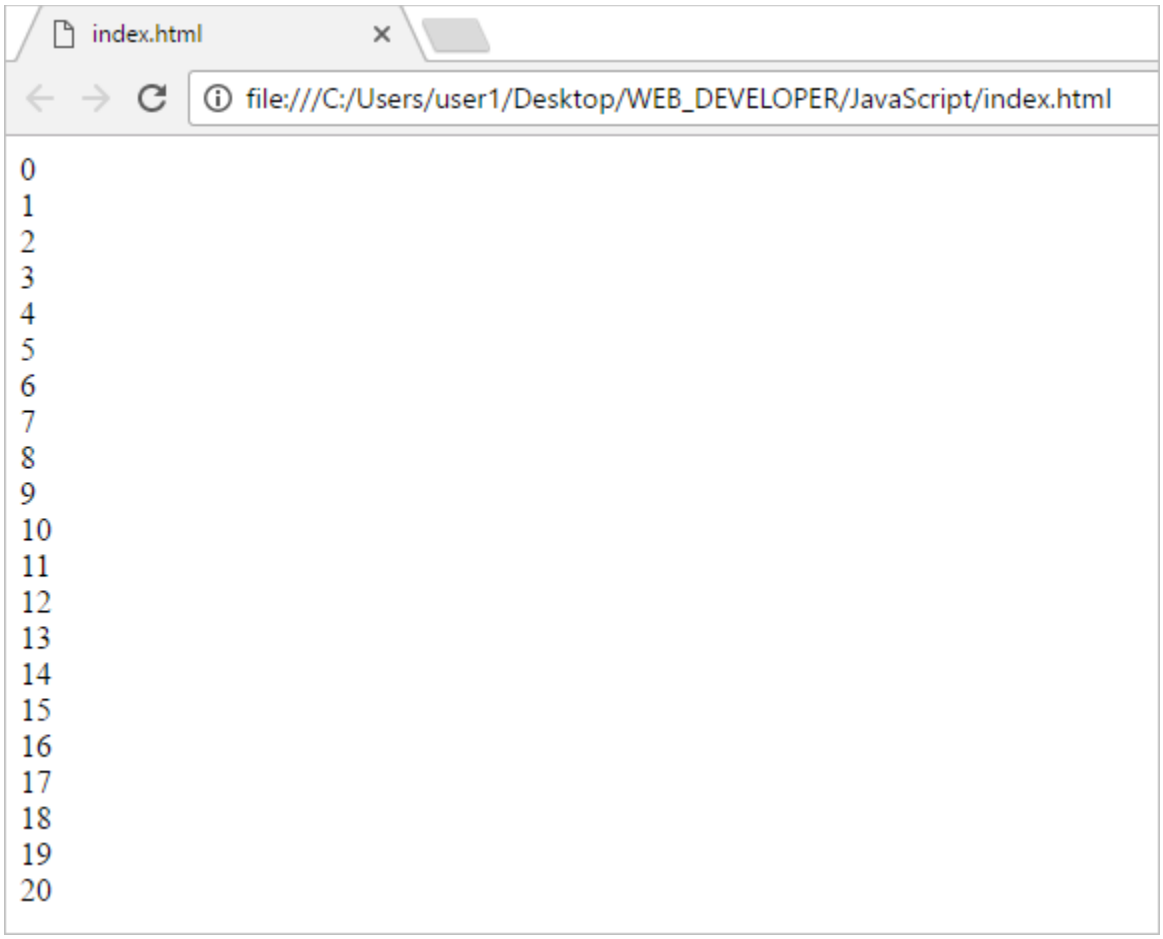
Sintaksa:

```
while (kushti) {
bllok i kodit
}
```

Kushti mund të jetë i vërtet ose i gabuar .

Marrim në konsideratë shembullin e mëposhtëm:

```
var i=0;
while (i<=20) {
document.write(i + "<br />");
i++;
}
```



Nese harrojmë të rrisin variablat me nga një atëhere, cikli nuk do të përfundojë asnjëherë.
Duhet të sigurohemi që kushti eventualisht do të bëhet false.

Cikli Do...While

Cikli **do...while** është një variant i ciklit while. Cikli do të ekzekutojë njëherë, do të ekzektujojë njëherë bllokun e kodit, **para** se të kontrollojë nëse kushti është i vërtetë, dhe më pas do të përsëritet derisa kushti të jetë i vërtetë.

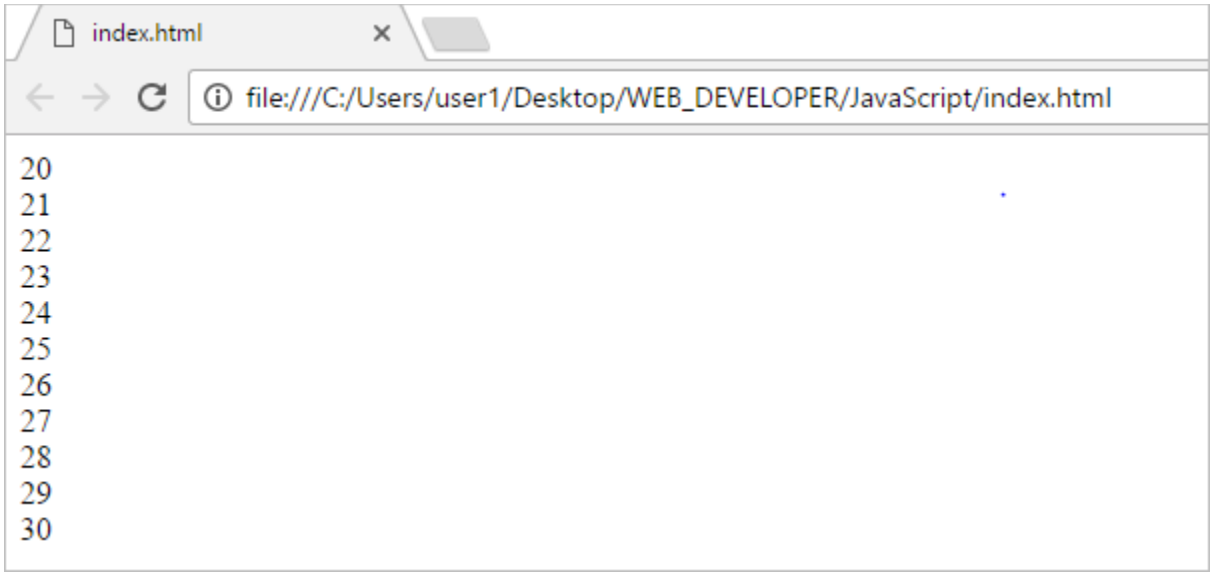
Sintaksa:

do {
blloku kodit
}
while (kushti);

Shembull:

```
var i=20;
do {
document.write(i + "<br />");
i++;
}
while (i<=30);
```

Rezultati:



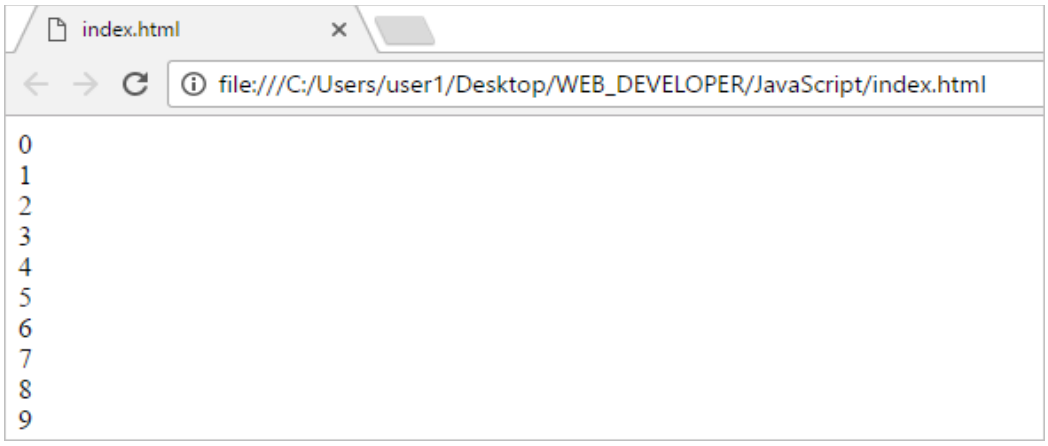
Cikli do të ekzekutohet të paktën njëherë, edhe pse kushti nuk është i vërtetë, sepse blloku kodit do të ekzekutohet para se kushti të jetë i vërtetë.

Break

Instruksioni **break** "kapërcen" kapërcen ciklin dhe vazhdon ekzekutimin pas ciklit.

```
for (i = 0; i <= 20; i++) {
if (i == 10) {
break;
}
document.write(i + "<br />");
}
```

Sapo arrin numrin 10, do të kapërcejë ciklin dhe do të ekzekutojë kodin pas ciklit.

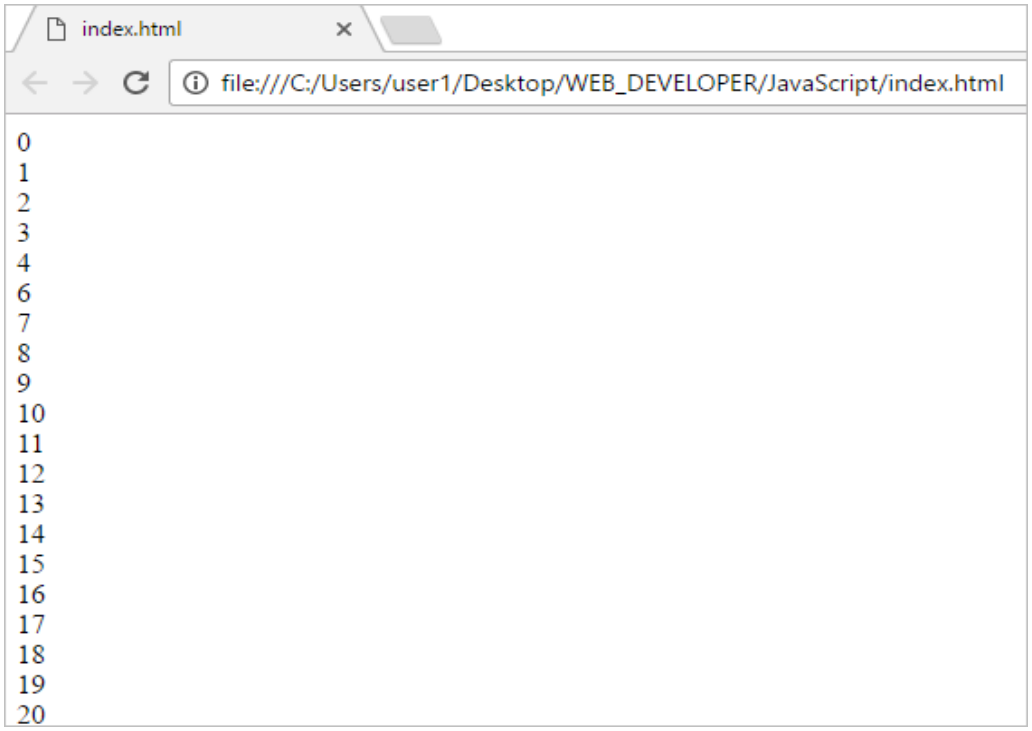


Continue

Instruksioni **continue** kapercen vetëm një cikël loop, dhe vazhdon me iteracionin tjetër.

```
for (i = 0; i <= 20; i++) {
if (i == 5) {
continue;
}
document.write(i + "<br />");
}
```

Rezultati:



Vlera 5 nuk është printuar, sepse indtruksioni **continue** e ka kapërcyer këtë cikël.

► Funksionet

Funksionet e përcaktuara nga përdoruesi

Një funksion javascript është një bllok kodi i dizenjuar i cili është projektuar për të kryer një detyrë të caktuar.

Avantazhet kryesore të përdorimit të funksioneve:
Ripërdorimi i kodit: Kodi përcaktohet njëherë dhe ripërdoret në shumë raste të tjera.Përdoret disa herë i njëjti kod me disa argumenta, për të marrë rezultate të ndryshme. Një funksion javaScript ekzekutohet kur thirret.

Përcaktimi i Funksioneve

Për të përcaktuar një funksion javascript, përdoret fjala kyçe **function** , ndjekur nga emri i funksionit, dhe **dy kllapat ()**.

Kodi i cili do të ekzekutohet nga funksioni do të jetë brenda kllapave {}.

```
function emri_funksionit() {
 //kodi që do të ekzekutohet
}
```

Emrat e funksioneve mund të përmbajnë shkronja, numra, underscores, dhe shenjën e dollarit (të njëjtat rregulla si tek deklarimi i variablave).

Thërritja e funksioneve

Për ekzekutimin e funksionit, duhet ta thërrisni diku tjetër në program.
Për të thërritur një funksion, fillohet me emrin e funksionit, ndjekur nga argumentat në kllapat gjarpërushe

Shembull:

```
function emriFunksionit() {
alert("Thërritja e një funksioni!");
}
```

```
emriFunksionit();
//Alert " Thërritja e një funksioni!");
```

Thërritja e një funksioni duhet të përfundoj gjithmonë me pikëpresje.

Thërritja e funksioneve

Funksioni përcaktohet vetëm njëherë, JavaScript ju lejon që ta thërrisni sa herë që dëshironi.

```
function emriFunksionit() {
alert("Alert box!");
}

emriFunksionit();
//"Dritarja alert!"

// disa kode të tjera

emriFunksionit();
//"Dritarja alert!"
```

Parametrat e funksioneve

Funksionet mund të marrin parametra.
Parametrat e funksioneve janë emrat e përcaktuar në thërritjen e funksionit.

Sintaksa:

```
emriFunksionit(param1, param2, param3) {
// kodi
}
```

Ashtu si variablat, parametrat duhet të jenë emra, të cilat duhet të ndahen me presje brenda kllapave.

Përdorimi i parametrave

Pas përcaktimit, parametrat mund të përdoren brenda funksionit.

```
function thuajPershendetje (emri) {
alert("Pershendetje, " + emri);
}

thuajPershendetje (Mira)
//Dritarja Alert "Përshendetje, Mira"
```

Funksioni merr një parameter, i cili quhet emri. Kur thërritet emri funksioni, vlera e parametrit vendoset brenda kllapave të funksionit.

Argumentat e funksionit janë vlerat reale që i kalohen ose merren nga funksioni

Parametrat e funksioneve

Ju mund të përcaktoni një funksion të vetëm, dhe të kaloni vlera parametrash të ndryshëm në të.

```
function thuajPershendetje(emri) {
alert("Përshendetje, " + emri);
}
thuajPërshendetje("Mira");
thuajPërshendetje("Beni");
```

Në rastin e mësipërm do të kemi ekzekutimin e kodit për secilin argument.

Parametrat shumëfishe

Për një funksion mund të përcaktoni më shumë se një parametër të ndarë me presje.

```
function emriFunksionit(x, y) {
// kodi
}
```

Shembulli i mësipërm përcakton **emriFunksionit(x, y)** i cili merr dy parametrat x dhe y.

Parametrat përdoren brenda përcaktimit të funksionit.

```
function Mosha(emri, mosha) {
document.write( emri + " është " + mosha + " vjeç.");
}
```

Kur thërritet funksioni, argumentat duhet të jepen në të njëjtën radhë siç janë përcaktuar.

```
function Mosha(emri, mosha) {
document.write( emri + " është " + mosha + " vjeç.");
}

Mosha("Mira", 20)
//Output-i "Mira është 20 vjeç."
```

Pas përcaktimit të funksionit, ju mund ta thërrisni sa herë të jetë e nevojshme.
Funksionet JavaScript nuk kontrollojnë numrine argumentave që marrin.
Nëse një funksion thërritet pa argumenta (ose më pak se sa janë deklaruar), vlerat që mungojnë do të njihen si të pa përcaktuar (**undefined)**, e cila tregon se variablës i është shenjuar një vlerë.

► Funksioni Return

Një funksion mund të ketë një instruksion opsional **return**. Përdoret për kthimin e një vlere nga funksioni.
Ky instruksion zakonisht është i dobishëm kur funksioni duhet të kthejë një vlerë. Kur JavaScript arrin një instruksion **return** funksioni ndalon së ekzekutuari.

Përdor instruksionin **return** për të kthyer një vlerë.
Për shembull, le të llogarisim prodhimin e dy numrave, dhe të kthehet rezultati.

```
function Prodhimi(a, b) {
return a * b;
}

var x = Prodhimi(5, 6);
// Vlera do t'i kthehet x
// x është e barabartë me 30
```

Nese nuk kthehet asgjë nga funksioni do të merret **undefined**.

Një shembull tjetër:

```
function Shuma (a, b) {
var c = a+b;
return c;
}
document.write( Shuma(40, 2) );
//Output-i 42
```

► Dritaret pop-up

JavaScript ofron tre lloje dritaresh pop-u, **Alert**, **Prompt**, dhe dritaren **Confirm**.

► Dritarja Alert

Një dritare **alert** përdoret kur ju duhet të siguroheni se informacioni u prezantua tek përdoruesi.
Kur shfaqet një dritare alert, përdoruesi duhet të klikojë OK për të vazhduar.
Funksioni **alert** merr një parametër, i cili është teksti i shfaqur në dritare.

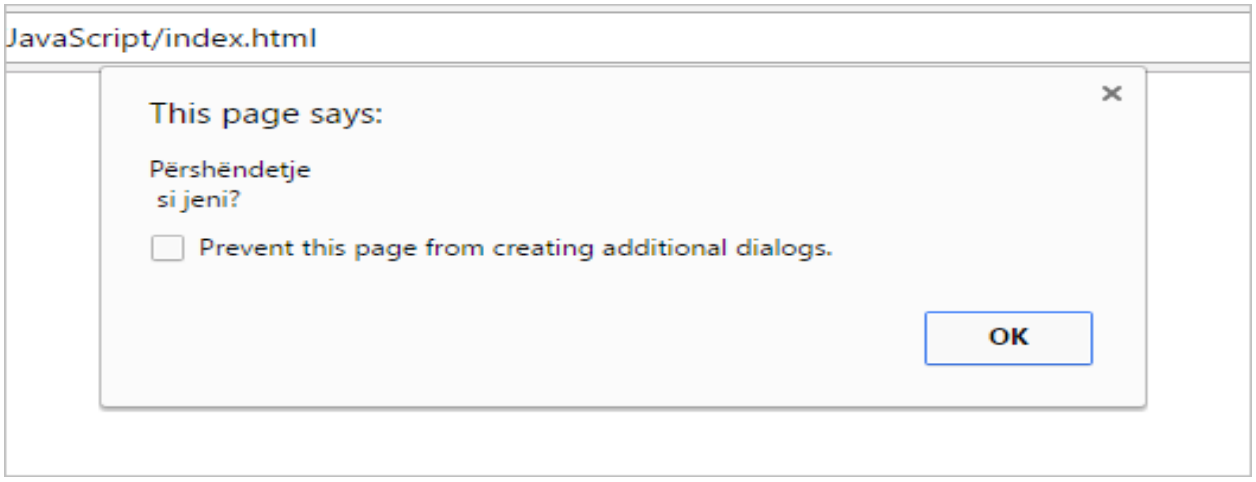
Shembull:

```
alert("Dëshironi të largoheni nga kjo faqe?");
```

Për të kaluar në një rresht të ri në një dritare pop-up përdoret back slash dhe n/

alert("Përshëndetje /**n** si jeni?");

Rezultati:



Duhet të jemi të kujdeshëm kur përdorim dritaren alert sepse përdoruesi mund të vazhdojë të përdorë faqen tuaj vetëm nëse klikohet OK.

► Dritarja Prompt

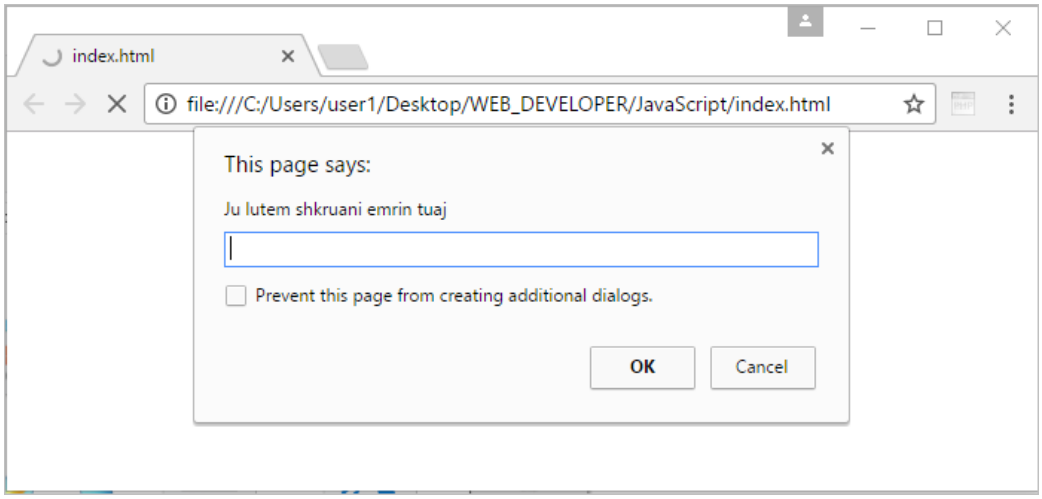
Dritarja **prompt** përdoret kur duhet të merren vlera nga përdoruesi para se të vazhdohet të punohet me faqen.
Kur një dritare prompt bën pop-up, përdoruesi duhet të klikoj tek OK ose Cancel për të vazhduar pas marrjes së vlerave input. Nëse përdoruesi klikon OK, dritarja kthen dritaren input. Nëse përdoruesi klikon Cancel, dritarja kthen **null**.

Metoda **prompt()** merr dy parametra.
- E para shfaq tekst.
- Një tekst opsional

Shembull:

```
var perdorues = prompt("Ju lutem shkruani emrin tuaj");
alert(perdorues);
```

Dritarja prompt shfaqet si:

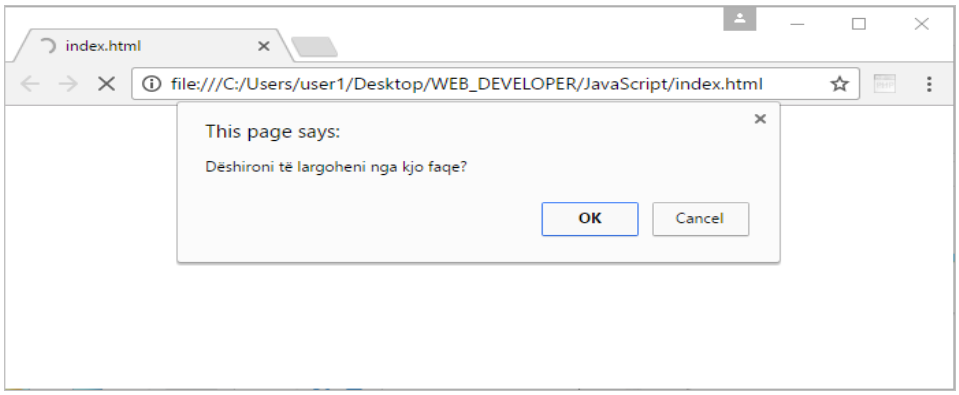


► Dritarja Confirm

Një dritare **confirm** zakonisht përdoret për të konfirmuar një tekst të caktuar për përdoruesin.
Kur shfaqet një dritare alert, përdoruesi duhet të klikojë OK për të vazhduar.
Nëse përdoruesi klikon OK, dritarja kthen **true**. Nëse përdoruesi klikon Cancel, dritarja kthen **false**.

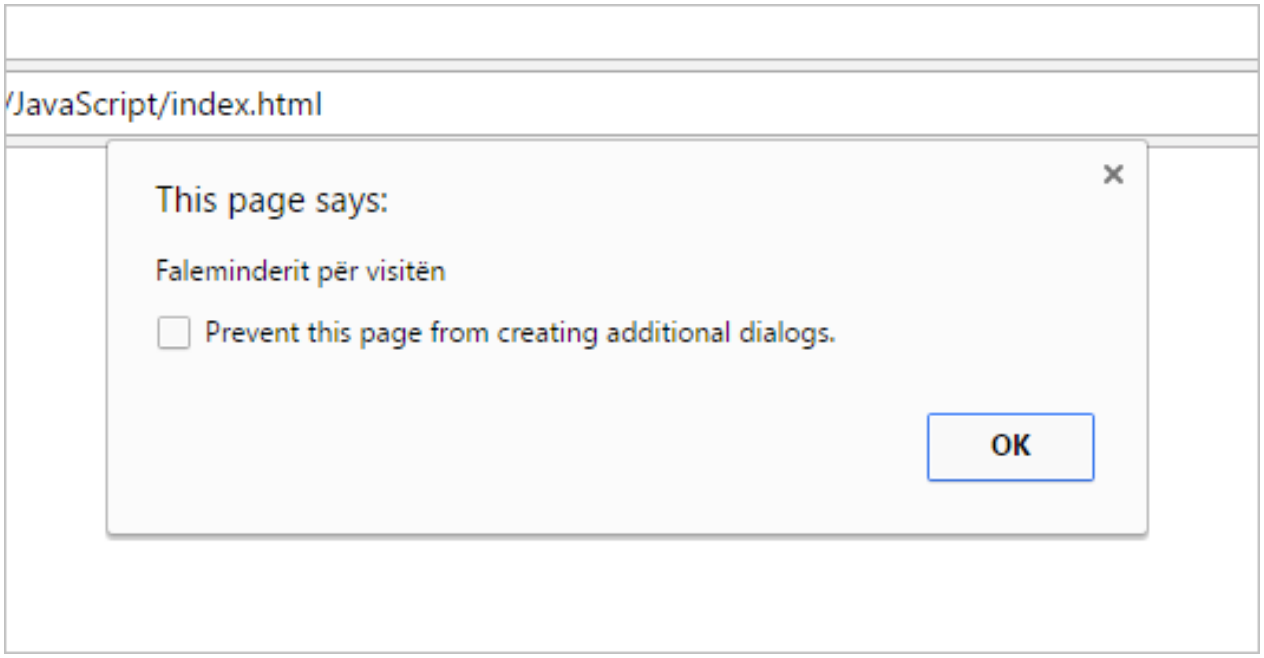
Shembull:

```
var rezultati = confirm("Dëshironi të largoheni nga kjo faqe?");
if (rezultati == true) {
alert("Faleminderit për visitën");
}
else {
alert("Faleminderit që zgjodhët të qëndroni me ne ");
}
```

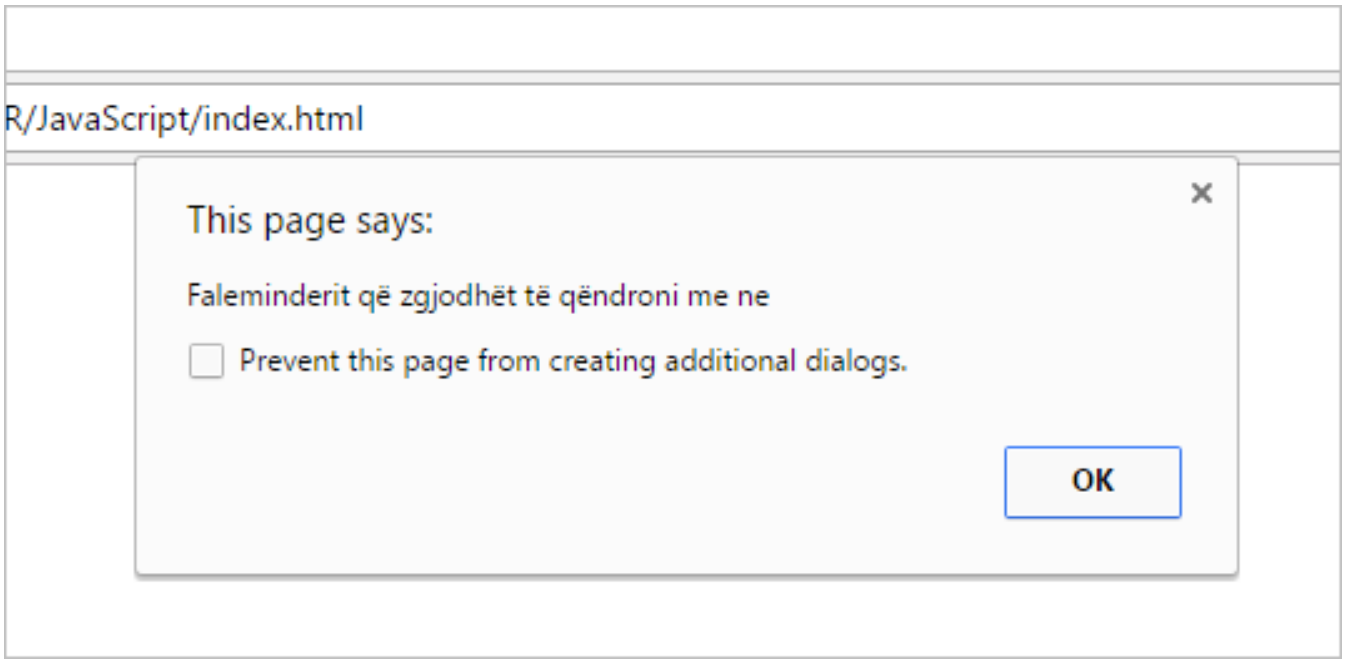


Rezultati:

Rezultati kur përdoruesi klikon **OK**:



Rezultati kur përdoruesi klikon **Cancel**:



► Objektet

Objektet JavaScript

Variablat në JavaScript janë mbajtës të të dhënave . Objektet janë variabla gjithashtu, por mund të përmbajnë shumë vlera.

Mendoni për objektet si vlera që janë shkruajtur si emra: çifte vlerash, me emrat dhe vlerat të ndara me kolona.
Shembull:

```
var person = {
emri: "Beni", mosha: 18,
shkolla: "informatik", gjatësia: 183
};
```

Këto vlera njihen si karakteristika.
Objektet JavaScript janë mbajtës të vlerave të emrave.

► Karakteristikat e objekteve

Ju mund t'i aksesoni karakteristikat e objekteve në dy mënyra.

emriObjektit.emriKarakteristikës
//ose
emriObjektit [' emriKarakteristikës ']
Shembulli më poshtë ilustron se si mund të aksesojmë moshen e objektit person.

```
var person = {
emri: "Beni", mosha: 18,
shkolla: "informatik", gjatësia: 183
};

var x = person.mosha;
var y = person['mosha'];
```

Karakteristika 'length' përdoret për të aksesuar numrin e karaktereve në string.

```
var  kursi = {emri: "JS", mesimi: objektet};
document.write(kursi.emri.length);
//Output-i  2
```

Metodat e objekteve

Një metodë objekti është një karakteristikë e cila mban përcaktimin e një funksioni.

Për të aksesuar metodën e një objekti përdoret sintaksa e mëposhtme:

```
emriObjektit.emriMetodës()
```

Për shembull, document.write() ka si output të dhëna. Funksioni write() është metodë e objektit document.

```
document.write("Përshëndetje!");
```

Metodat janë funksione të cilat ruhen si karakteristika të objekteve.

Konstruktori i Objektit:

Në shembullin e mësipërm ne krijuam një object sipas sintaksës së mëposhtme:

```
var person = {
emri: "Beni", mosha: 18 , shkolla: "informatikë"
};
```

Kjo ju lejon të krijoni një objekt të vetëm. Mënyra standarte për kriimin e "tipeve të objekteve" është të përdorim funksionin e konstruktorit të një objekti.

```
function person(emri, mosha, shkolla) {
this.emri = emri;
this.mosha = mosha;
this.shkolla = shkolla;
}
```

Funksioni i mësipërm është një konstruktor object (person), i cili merr parametrat dhe ja kalon karakteristikave të objektit. Fjala kyçe this ireferohet objektit aktual.
Shënim: this nuk është variabel. **This** është një fjalë kyçe vlera e së cilës nuk mund të ndryshohet.

Krijimi i objekteve

Kur kemi konstruktorin e një objekti, mund të përdorim fjalën kyçe new për krijimin e objekteve të reja të të njëjtit tip.

```
var p1 = new person("Beni", 18, "informatikë");
var p2 = new person("Mira", 17, "informatikë");

document.write(p1.mosha); // Outputs 18
document.write(p2.emri // Outputs "Mira"
```

p1 dhe *p2* janë objekte të tipit person. Karakteristikave janë dhënë vlerat përkatëse.

Merrni në konsideratë shembujt e mëposhtëm.

```
function person (emri, mosha) {
this.emri = emri;
this.mosha = mosha;
}
var Beni = new person("Beni", 18);
var Mira = new person("Mira", 17);
```

Aksesimi i karakteristikave të objektit duke përdorur sintaksën e pikës.

Emri i objektit	Emri i karakteristikës
Beni	. emri
Beni	. mosha
Mira	. emri
Mira	. mosha

► Inicializimi i objekteve

Hapsirat dhe thyerjet e rreshtave nuk janë të rëndësishëm. Përcaktimi I një funksioni mund të jetë me disa rreshta.

```
var Mira = {
emri: "Mira",
mosha: 18
};
var Beni = {
emri: "Beni",
mosha: 17
};
```

Nuk ka rëndësi se si është krijuar objekti, sintaksa e krijimit të karakteristikave dhe metodave nuk ndryshon.

```
document.write(Mira.mosha);
//Outputi 18
```

Objektet

Shtimi i Metodave

► Metodat

Metodat janë funksione që ruhen si karakteristika të objektit.

Sintaksa e mëposhte tregon krijimin e një objekti të metodës:

```
emriMetodës : funksioni() { rreshtat kod }
```

Aksesimi i metodës së objektit duke përdorur sintaksën e mëposhtme:

```
emriObjektit.emriMetodes()
```

Një metodë është një funksion që i përket një objekti. I referohemi duke përdorur fjalën kyçe this. Fjala Kyçe this përdoret si referencë në objektin aktual, që do të thotë ju mund të aksesoni karakteristikat e objektit dhe metodave duke e përdorur atë. Përcaktimi i metodave bëhet brenda konstruktorit të funksionit.

Për shembull:

```
function person(emri, mosha) {
this.emri = emri;
this.mosha = mosha;
this.ndryshoEmer = function (emri) {
this.emri = emri;
}
}

var p = new person("Beni", 21);
```

```
p.ndryshoEmer("Beni");
//Tani p.emri është e barabartë me "Beni"
```

Në shembullin e mësipërm, është përcaktuar një metodë e quajtur **ndryshoEmer** për funksionin person, i cili merr një parametër i cili quhet emri dhe ja shenjon karakteristikës së objektit**.**
Metoda **ndryshoEmer** ndryshon karakteristikën e objektit të emrit.

Metodat

Funksionet mund të përcaktohen edhe jashtë kondtruktorit të funksionit shoqëruar me:

```
object.function person(emri,mosha) {
this.emri= emri;
this.mosha = mosha;
this.ditelindja = vitiLindjes;
}
function vitiLindjes() {
return 2016 - this.ditelindja;
}
```

Siç e shikoni, ne i kemi dhënë karakteristikën e objektit të vititLindjes tek funksioni vitiLindjes. Fjala kyçe **this** përdoret për aksesimin e karakteristikës mosha të objektit, e cila thërret metodën.

Thërritja e zakonshme e funksionit.

```
function person(emri, mosha) {
this.emri= emri;
this.mosha = mosha;
this.ditelindja = vitiLindjes;
}
function vitiLindjes() {
return 2016 - this.mosha;
}

var p = new person("A", 26);
document.write(p.vitiLinjes());
// Output-i 1990
```

Thërritja e metodës me emrin e karakteristikës që specifikohet në funksionin e konstruktorit.

► Matricat JavaScript

Matricat ruajnë shumë vlera në një variabël të vetme.
Për të ruajtur emrat e kurseve ju duhet të përdorni tre variabla.

```
var kursi1 ="HTML";
var kursi2 ="CSS";
var kursi3 ="JS";
```

Por nëse keni 100 kurse? Zgjidhja është përdorimi i matricave.

```
var kurset = new Array("HTML", "CSS", "JS");
```

Kjo sintaksë deklaron një matricë me emrin kurset, e cila ruan tre vlera ose elemente.

► Aksesimi i një matrice

Në matricë i referohemi elementit të një matricë me anë të numrit të indeksit të shkruajtur në kllapa katrore.

Ky instruksion akseson vlerën e elementit të parë në matricën **kurset** dhe ndryshon vlerën e elementit të dytë.

```
var kurset = new Array("HTML", "CSS", "JS");
var kursi = kurset[0]; // HTML
kurset[1] = "C++"; //Ndryshon elementin e dytë
```

[0] është elementi i parë në matricë. [1] është i dyti. Indeksi i matricave fillon me **0**.
Aksesimi i një indeksi jashtë matrice, kthen vler/n **undefined**.

```
var kurset = new Array("HTML", "CSS", "JS");
document.write(kurset[10]);
//Output-i "undefined"
```

Matrica jonë kurset ka tre elementë, pra indeksi i 10-të , i cili është elementi i 11-të , nuk ekziston (është undefined).

► Krijimi i matricave

Matricat mund të deklarohen , duke thën numrin e elementëve që do të ruhen në fillim, dhe duke shtuar elementët më vonë.

```
var kurset = new Array(3);
kurset[0] = "HTML";
kurset[1] = "CSS";
kurset[2] = "JS";
```

Një matricë është një tip special i një objekti. Një matricë përdor **numrat për aksesimin e elementëve**, dhe një object përdor emrat për aksesimin e anëtarëve.

Matricat JavaScript janë dinamike,pra ju mund të deklaroni një matricë pa kaluar ndonjë argument me konstruktorin Array(), mund të shtohen më pas në mënyrë dinamike.Mund të shohen sa elementë të jetë e mundur.

```
var kurset = new Array();
kurset[0] = "HTML";
kurset[1] = "CSS";
kurset[2] = "JS";
kurset[3] = "C++";
```

► Array Literal

Për thjeshtësi, rilexueshmëri, dhe shpejtësi ekzekutimi, matricat mund të deklarohen duke përdorur sintaksën e **array literal**.

```
var kurset = ["HTML", "CSS", "JS"];
```

Karakteristika length

Matricat JavaScript kanë karakteristika dhe metoda të paracaktuara.
Karakteristika **length** e matricave kthen numrin e elementev të matricës.

```
var kurset = ["HTML", "CSS", "JS"];
document.write(kurset.length);
//Output-i 3
```

Vlera e karakteristikës **length** është gjithmonë një më shumë se indeksi i matricës .
Nëse matrica është bosh, karakteristika length kthen **0**.

► Kombinimi i matricave

Metoda JavaScript's **concat()** ju lejon të krijoni një matricë të re nga bashkimi i dy matricave.

Shembull:

```
var c1 = ["HTML", "CSS"];
var c2 = ["JS", "C++"];
var kurset = c1.concat(c2);
```

Matrica **kurset** rezulton me katër elementë (HTML, CSS, JS, C++). Veprimi **concat** nuk ndikon tek matricat *c1* dhe *c2* , thjesht kthen matricën e re të bashkuar.

Matricat shoqëruese

Shumë gjuhë programimi suportojnë matricat e emërtuara me indekse (tekst në vend të numbrave), të quajtura matrica shoqëruese.

Shembull:

```
var person = []; //matrice bosh
person["emri"] = "Beni";
person["mosha"] = 26;
document.write(person["mosha"]);
//Output-i "26"
```

Tani, person është trajtuar si objekt, në vend të matricës.
Indekset e emërtuara "emri" dhe "mosha" bëhen karakteristika të objektit person.

Mbani mend që JavaScript **nuk** suporton matricat me indekse të emërtuara.
Në JavaScript, matricat përdorin gjithmonë indekse të numërtuara.

► Objektet matematikore

Objektet matematikore ju lejojnë të kryeni veprime matematikore, dhe përfshijnë shumë karakteristika.

Karakteristika	Përshkrimi
E	Konstantja e Eulerit
LN2	Logaritmi natyror i 2
LN10	Logaritmi natyror i 10-tës
LOG2E	Logaritmi me bazë 2 i konstantes së Eulerit
LOG10E	Logaritmi me bazë 10 i konstantes së Eulerit
PI	Rikthen konstanten PI

Për shembull:

```
document.write(Math.PI);
//Output-i 3.141592653589793
```

Veprimet matematikore nuk kanë konstruktorë. Nuk është e nevojshme të krijohet një objekt në fillim.

Metodat e objekteve matematikore

Objektet matematikore përmbajnë një numër të metodave që përdoren për llogaritje. Për shembull, llogaritja e rrënjës katrore.

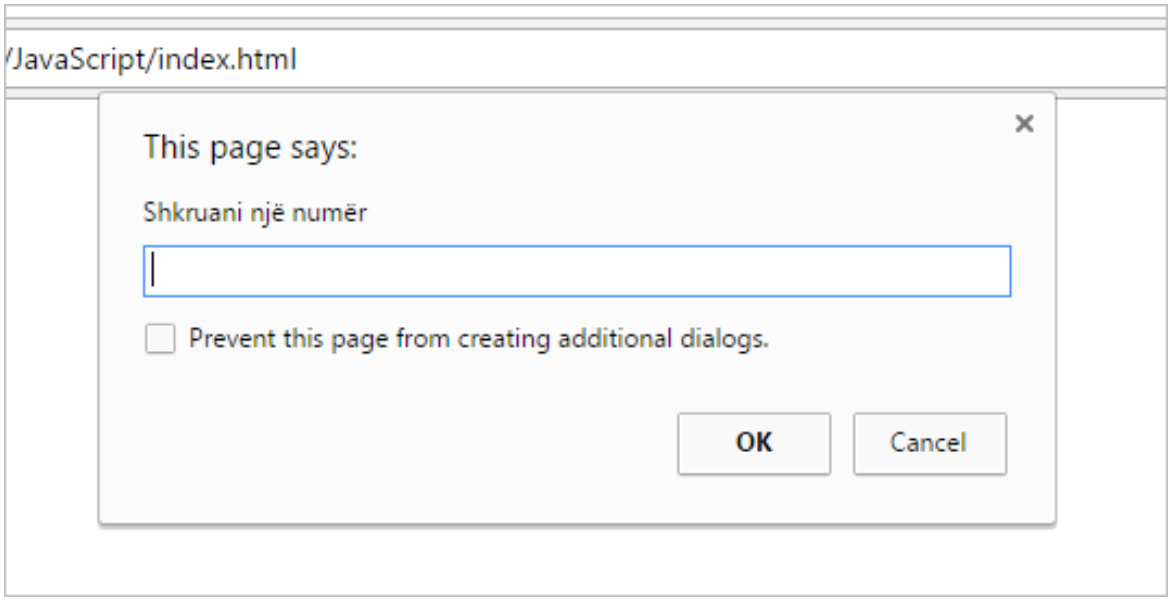
```
var numbër = Math.sqrt(4);
document.write(numbër);
//Output-i 2
```

Objektet matematikore

Shembulli më poshtë pyet përdoruesin të japë një numbër input dhe afishon rrënjën katrore të tij.

```
var n = prompt("Shkruani një numër", "");
var përgjigja = Math.sqrt(n);
alert("Rrënja katrore e " + n + " është" + përgjigja);
```

Rezultati:



Metoda setInterval

Metoda **setInterval()** thërret një funksion që specifikohet në milisekonda, derisa thërritet metoda **clearInterval()** .

Shembull:

```
function intervali() {
alert("Përshëndetje");
}
```

```
setInterval(intervali, 4000);
```

Funksioni intervali do të thërritet çdo 4 sekonda (1000 ms = 1 sekond).

Objekti Date

Objekti **Date** bën të mundur punën me datat.

Një datë konsiston në vit, muaj, dita, ora, miutat, sekondat, dhe milisekondat.

Duke përdorur **new Date()**, krijojmë një objekt të ri të datës me datën dhe kohën aktuale .

```
var d = new Date();
//d ruan datën dhe orën aktuale
```

Mënyrat e tjera të inicializimit të datës lejojnë krijimin e një objekti të ri të datës për një datë specifike.

```
new Date(milisekonda)
new Date(dateString)
new Date(viti, muaji, dita, orët, minutat, sekondat, milisekondat)
```

Metoda Data

Kur krijohet një objekt date, ekzistojnë një numbër metodash me të cilat mund të kryhen veprime.

Metoda	Përshkrimi
getFullYear()	Kthen vitin
getMonth()	Kthen muajin
getDate()	Kthen datën
getDay()	Kthen ditën
getHours()	Kthen orën
getMinutes()	Kthen minutat
getSeconds()	Kthen sekondat
getMilliseconds()	Kthen milisekondat

Shembull:

```
var d = new Date();
var hours = d.getHours();
//ora është e barabartë me orën aktuale
```

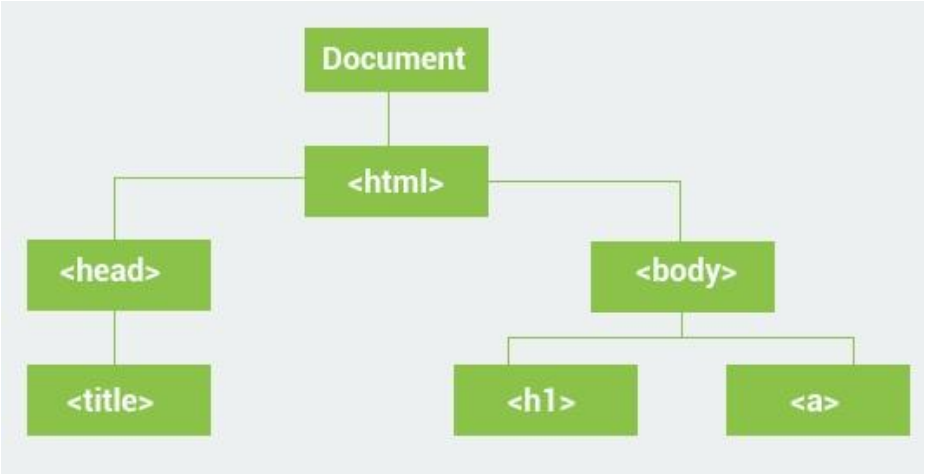
Programi më poshtë afishon çdo second orën aktuale.

```
function printoOren() {
var d = new Date();
var ora = d.getHours();
var minuta = d.getMinutes();
var sekonda = d.getSeconds();
document.body.innerHTML = ora+":"+minutat+":"+sekonda;
}
setInterval(printoOren, 1000);
```

► DOM & Eventet

DOM

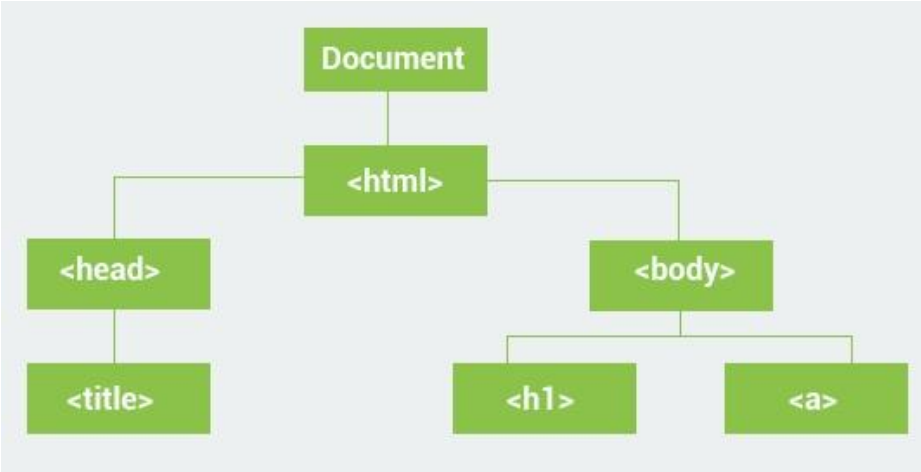
Kur ju hapni një faqe në browser, ngarkohet HTML-ja e faqes. Për të realizuar këtë, browser-i ndërton **Document Object Model** të kësaj faqeje, i cili është një model i orientuar nga objektet të strukturës logjike.
DOM e një dokumenti HTML mund të prezantohet si kuti të folezuara:



JavaScript mund të manipulojë DOM e një faqe në mënyrë dinamike për të shtuar, fshirë apo modifikuar elementet.

► Pema DOM

DOM riprezanton një dokument si strukturë peme.
Elementët HTML janë nyjet e kësaj peme.
Të gjitha këto nyjet kanë një lidhje logjike me njëra-tjetrën.
Nyjet kanë nyje fëmijë. Nodes në të njëjtën nivel në pemë quhen **siblings**.
Për shembull, marrim në konsideratë strukturën e mëposhtme:



Për shembullin e mësipërm:

<html> ka dy fëmijë (<head>, <body>);
<head> ka një fëmijë (<title>) dhe një prind (<html>);
<title> ka një prind (<head>) dhe asnjë fëmijë;
<body> ka dy fëmijë (<h1> dhe <a>) dhe një prind (<html>);

Është shumë e rëndësishme të kuptohet lidhja midis elementeve të një dokumenti HTML në mënyrë që t'i manipulojmë me anë të JavaScript.

► Objekti dokument

Në JavaScript është një objekt dokument i paracaktuar, i cili mund të përdoret për aksesimin e të gjithë elementev në DOM.
Me fjalë të tjera, objekti dokument është zotëruesi (ose rrënja) e të gjithë objekteve në një faqe web.
Pra, nëse doni të aksesoni objektet në një faqe HTML, gjithmonë fillohet duke aksesuar objektin dokument.

Shembull:

```
document.body.innerHTML = "teksti";
```

Ku **body** është një element DOM, ne mund ta aksesojmë objektin **document** dhe të ndryshojmë përmbajtjen e karakteristikës **innerHTML**.
Karakteristika **innerHTML** mund të përdoret pothuajse në të gjithë elementët HTML për të ndryshuar përmbajtjen e tyre.

► Selektimi i elementëve

Të gjithë elementët HTML janë **objekte**. Siç e dimë çdo objekt ka karakteristika dhe metoda.
Objekti **document** ka metoda të cilat ju lejonë të selektoni elementin e duhur HTML.
Këto tre metoda më të përdoruara për të selektimin e elementev HTML:

```
//sipas id
document.getElementById(id)

//sipas emrit të klasës
document.getElementsByClassName(emriKlases)

//sipas tagut HTML
document.getElementsByTagName(emri)
```

Në shembullin më poshtë, metoda getElementById përdoret për selektimin e elementit me id="demo"dhe ndryshon përmbajtjen:

```
var elem = document.getElementById("demo");
elem.innerHTML = "Përshendetje!";
```

► Ndryshimi i atributeve

Sapo të selektoni elementët me të cilët doni të punoni, mund të ndryshoni atributet e tyre.Ne mund të ndryshojmë përmbajtjen e një elementi në HTML duke përdorur karakteristikën **innerHTML**.
Në mënyrë të ngjashme mund të ndryshojmë atributet e elementëve.

Për shembull, ne mund të ndryshojmë atributin **src** të një imazhi:

```
<img id="imazh" src="logo1.png" alt="" />
<script>
var el = document.getElementById("imazh");
el.src = "logo2.png";
</script>
```

Ne mund të ndryshojmë atributin **href** të një linku:

```
<a href="http://www.shembull.com">kliko ketu</a>
<script>
var el = document.getElementsByTagName("a");
el[0].href = "http://www.google.com";
</script>
```

Praktikisht të gjithë elementët HTML mund të ndryshohen duke përdorur JavaScript.

Ndrsyhimi i stilit

Stili i elementeve HTML mund të ndryshohet gjithashtu me anë të JavaScript.
Të gjithë atributet e stilit mund të akseohen duke përdorur objektin style të një elementi.

Shembull:

```
<div id="demo" style="width:200px">Teksti </div>
<script>
var x = document.getElementById("demo");
x.style.color = "6600FF";
x.style.width = "100px";
</script>
```

Kodi i mësipërm ndryshon ngjyrën e tekstit dhe gjerësinë e elementit div. Të gjithë karakteristikat CSS mund të vendosen ose modifikohen duke përdorur JavaScript. Emrat e karakteristikave përdoren pa vijë në mes, për shembull: karakteristika **background-color** do të referohet si **backgroundColor**.

► Krijimi i elementëve

Përdorni metodat e mëposhtme për krijimin e nyjeve të reja:
element.**cloneNode**() klonon elementin dhe kthen nyjen rezultat.
document.**createElement**(element) krijon një nyje elementi të re.
document.**createTextNode**(tekst) krijon një nyje të re tekst.

Shembull:

```
var nyje = document.createTextNode("Tekst");
```

Shembull:

```
<div id ="demo">përmbajtja</div>
<script>
//krijimi i një paragrafi të ri
var p = document.createElement("p");
var nyje = document.createTextNode("teksti");
//shtimi i tekstit në paragraf
p.appendChild(nyje);
var div = document.getElementById("demo");
//shtimi i një paragrafi në div
div.appendChild(p);
</script>
```

Ri-zhvendosja e elementeve

Për të rizhvendosur një element HTML, ju duhet të selektoni elementin prind të një elementi duke përdorur metodën **removeChild**(nyje).

Shembull:

```
<div id="demo">
<p id="p1">Ky është paragraf.</p>
<p id="p2">Ky është një paragraf tjetër.</p>
</div>

<script>
var prind = document.getElementById("demo");
var femij = document.getElementById("p1");
prind.removeChild(femij);
</script>
```

Kjo zhvendos elementin me id="p1" nga faqja.

► Animacionet

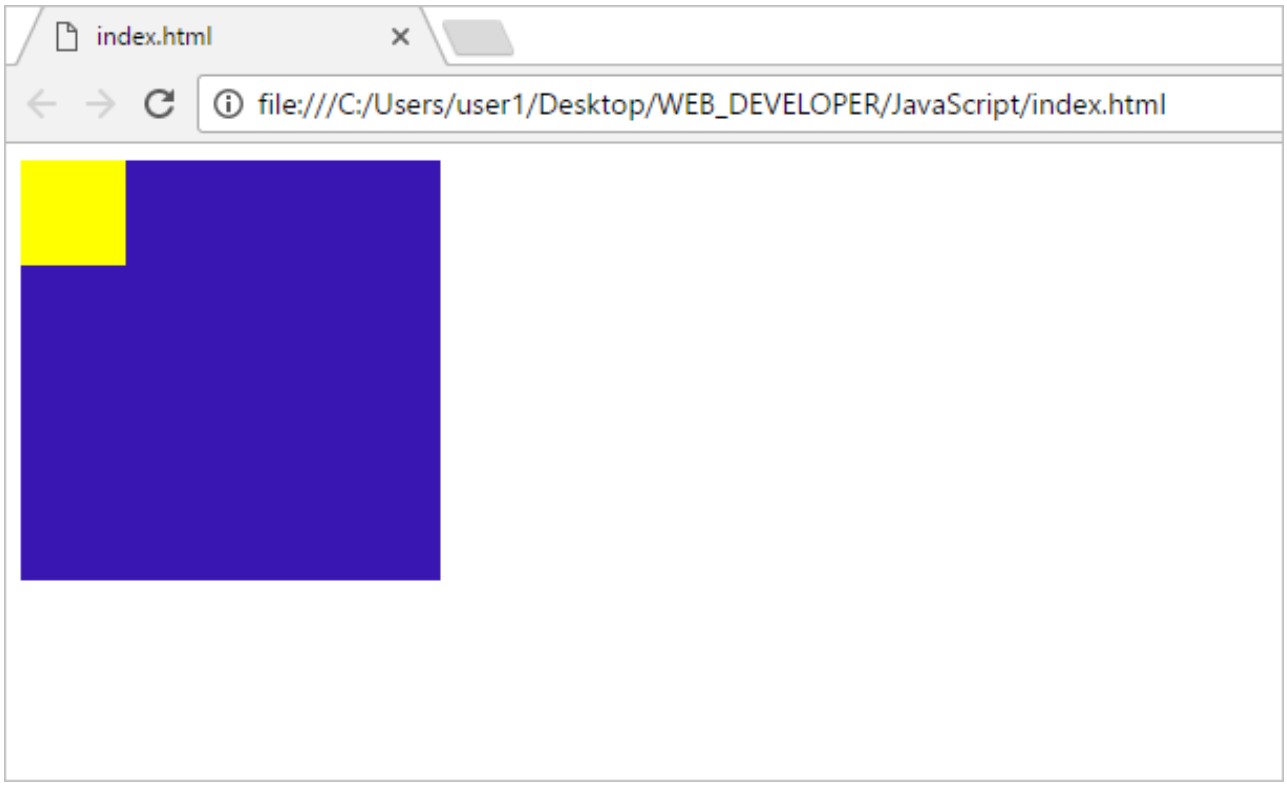
Tani që dimë se si të selektojmë dhe ndryshojmë elementët DOM mund të krijojmë një animacion të thjeshtë.

Le të krijojmë një faqe të thjeshtë HTML me elementin **box** i cili do të animohet duke përdorur JS.

```
<style>
#container {
width: 200px;
height: 200px;
background: #3915b2;
position: relative;
}
#box {
width: 50px;
```

```
height: 50px;
background: #ffff00;
position: absolute;
}
</style>
<div id="container">
<div id="box"> </div>
</div>
```

Elementi **box** ndodhet brenda elementit **container**. Atributi pozicion i elementeve: container është **relative** dhe për box është **absolute**.



Do të krijojmë animacionin e lëvizjes së kutisë brenda containerit, do të ndryshojmë karakteristikat e një elementi duke i ndarë në interval kohe. Kjo mund të duke përdorur metodën **setInterval**(), e cila lejon krijimin e një kohëzuesi dh ethërritjen e një funksioni për ndryshimin e karakteristikave në mënyrë të përsëritur.

Shembull:

```
var t = setInterval(leviz, 500);
```

Kodi i mësipërm krijon një kohëzues i cili thërret funksionin leviz() çdo 500 milisekonda. Tani përcaktojmë funksionin **leviz**(), e cila ndryshon pozicionin e kutisë.

```
// pozicioni fillestar
var pos = 0;
//elementi box
var box = document.getElementById("box");

function leviz() {
pos += 1;
box.style.left = pos+"px"; //px = pixels
}
```

Funksioni **leviz**() inkrementon karakteristikën **left** të elementit box me nga një sa herë thërritet.

Kodi i mëposhtëm përcakton një timer i cili thërret funsksionin leviz() çdo 10 milisekonda :
var t = setInterval(leviz, 10);
Megjithatë, kjo e bën box të lëviz përgjithmonë djathtas. Për të ndaluar animacionin kur box arrin fundin e container, ne shtuam metodën **clearInterval**() për ta ndaluar.

```
timer.function leviz() {
if(pos >= 150) {
clearInterval(t);
}
else {
pos += 1;
box.style.left = pos+"px";
}
}
```

Kur atributi i majtë i box arrin vlerën 150, box arrin fundin e container, bazuar në gjerësinë e container prej 200 dhe gjerësinë e box prej 50.

```
var pos = 0;
//elementi box
```

```
var box = document.getElementById("box");
var t = setInterval(leviz, 10);
function leviz() {
if(pos >= 150) {
clearInterval(t);
}
else {
pos += 1;
box.style.left = pos+"px";
}
}
```

► Eventet

Në JavaScript mund të shkruhen kode që mund të ekzekutohen kur evente të caktuara ndodhin, si për shembull kur një përdorues klikon një element HTML, lëviz mousin apo plotëson një formular. Eventet e zakonshme HTML përfshijnë:

Eventet	Përshkrimi
onclick	Ndodh kur përdoruesi klikon një element
onload	Ndodh kur ngarkohet një objekt
onunload	Ndodh kur nuk ngarkohet faqja
onchange	Ndodh kur ndryshohet përmbajtja e një elementi në kontakt forme (<input>, <keygen>, <select>, <textarea>)
onmouseover	Ndodh kur mousi kalon mbi elementin
onmouseout	Ndodh kur mousi kalon jashtë një elementi
onmousedown	Ndodh kur klikohet një buton në elementin
onmouseup	Ndodh kur klikohet një buton mbi elementin
onblur	Ndodh kur një element humbet fokusin
onfocus	Ndodh kur një element merr fokus

Eventet shtohen brenda tageve HTML si atribute.
Për shembull:

```
<p onclick="funksioni()">teksti</p>
```

► Menaxhimi i eventeve

Shfaqja e një dritareje alert popup kur përdoruesi klikon një buttob të caktuar:

```
<button onclick="show()"> Kliko këtu </button>
<script>
function show() {
alert("Përshendetje");
}
</script>
```

Event handler mund t'i shenjohet një funksioni .

Për shembull:

```
var x = document.getElementById("demo");
x.onclick = function () {
document.body.innerHTML = Date();
}
```

► Eventet

Eventet **onload** dhe **onunload** ndodhin kur përdoruesi ngarkon apo nuk ngarkon faqen. Përdoren për performimin e veprimeve para se të ngarkohet faqja.

```
<body onload="funksioni ()">
```

Në mënyrë të ngjashme, eventi **window.onload** mund të përdoret për ekzekutimin e kodit pas ngarkimit të gjithë faqes.

```
window.onload = funksioni() {
//kodi
}
```

Eventi **onchange** përdoret gjerësisht në kutitë e tekstit. Eventi thërritet kur teksti brenda kutisë ndryshon dhe fokusi humbet nga ky element:

Për shembull:

```
<input type="text" id="emri" onchange="ndrysho()">
<script>
function ndrysho() {
var x = document.getElementById("emri");
x.value= x.value.toUpperCase();
}
</script>
```

► Event Listener

Metoda **addEventListener**() i bashkangjit një event një elementi pa mbi shkruar eventet ekzistuese. Një elementi mund t'i shtohen disa evente.

► Image Slider

Krijimi i një slideri imazhesh në JavaScript. Imazhet do të ndryshohen duke përdorur butonat "Next" dhe "Prev".
Tani, le të krijojmë HTML, e cila përfshin një imazh dhe dy butona navigimi.

```
<div>
<button> Prev </button>
<img id="slider" src="1.jpg"
width="200px" height="100px"/>
<button> Next </button>
</div>
```

Përcaktojmë imazhet në një matricë:

```
var imazhet= [
" 1.jpg",
" 2.jpg",
" 3.jpg"
];
```

Mund të përdoret një numër çfarëdo imazhesh.

Tani duhet të menaxhojmë klikimet e buttonave Next dhe Prev dhe thërritjen e funksioneve përkatëse të cilat do të bëjnë ndryshimin e imazheve:

HTML:

```
<div>
<button onclick="prev()"> Prev </button>
<img id="slider" src="1.jpg"
width="200px" height="100px"/>
<button onclick="next()"> Next </button>
</div>
```

JS:

```
var imazhet = [
"1.jpg",
"2.jpg",
"3.jpg"
];
var num = 0;
function next() {
var slider = document.getElementById("slider");
num++;
if(num >= imazhet.length) {
num = 0;
}
slider.src = imazhet[num];
}

function prev() {
var slider = document.getElementById("slider");
num--;
if(num < 0) {
num = imazhet.length-1;
}
slider.src = imazhet[num];
}
```

Variablat **num** mban imazhin aktual. Klimet e butonave next dhe previous menazhohen nga funksionet korresponduese, e cila ndryshon burimin e imazhit në një nga elementët e matricës.

► Validimi i formave

HTML5 shton disa attribute të cilat lejojnë validimin e formave. Për shembull, atributi **requires** mund të shtohet tek një fushë input për të kërkuar me patjetër plotësimin e asaj forme. Validimet më komplekse të formave mund të bëhet duke

përdorur JavaScript. Tagu form ka një event **onsubmit** i cili mund të përdoret për menaxhimin e validimit. Për shembull, le të krijojmë një form me dy inpute dhe një button. Teksti në të dy fushat duhet të jetë i njëjtë dhe jo bosh për të kaluar validimin.

```
<form onsubmit="return validate()" method="post">
Numri: <input type="text" name="num1" id="num1" />
<br />
Përsërit: <input type="text" name="num2" id="num2" />
<br />
<input type="submit" value="Submit" />
</form>
```

Tani përcaktojmë funksionin **validate**() :

```
funksioni validate() {
var n1 = document.getElementById("num1");
var n2 = document.getElementById("num2");
if(n1.value != "" && n2.value != "") {
if(n1.value == n2.value) {
return true;
}}
alert("Vlerat duhet të jenë të njëjta dhe jo boshe");
return false;
}
```

Forma nuk do të dërgohet nëse eventi **onsubmit** do të kthejë **false**.