



erasmus

HOGESCHOOL BRUSSEL

Web Development Advanced

A large yellow circle is centered on the page. Inside the circle, the letters 'JS' are written in a bold, dark gray, sans-serif font.

JS

LINKEN NAAR EEN JAVASCRIPT-BESTAND VANUIT EEN HTML-PAGINA

```
<DOCTYPE html>
<html>
  <head>
    <title>Constructive & Co</title>
    <link rel="stylesheet" href="c01.css" />
  </head>
  <body>
    <h1>Constructive & Co</h1>
    <p>For all orders and enquiries,
    please call <em>555-3344</em></p>
    <script src="js/add-content.js"></script>
  </body>
</html>
```

JAVASCRIPT
WORDT UIGEVOERD
WAAR HET WORDT
GEVONDEN IN DE
HTML

Wanneer de browser een
`<script>` element tegenkomt,
zal hij:

Stoppen om het script te laden
Controleren of hij een actie moet
uitvoeren

VARIABELEN

DECLAREREN VAN EEN VARIABLE


```
var quantity;
```



A horizontal line with vertical end caps is positioned below the word 'var'. A short vertical line descends from the center of this horizontal line to the word 'KEYWORD'.

KEYWORD

```
var quantity;
```

VARIABLE NAME

TOEKENNEN VAN EEN WAARDE AAN EEN VARIABELE

```
quantity = 3;
```



VARIABLE NAME

```
quantity = 3;
```

ASSIGNMENT OPERATOR

```
quantity = 3;  
          |  
        VALUE
```

ARRAYS

```
var colors = ['pink', 'yellow', 'green'];
```



```
var colors = ['pink', 'yellow', 'green'];
```

```
colors[ ];
```

```
var colors = ['pink', 'yellow', 'green'];
```

```
colors[0];
```

```
var colors = ['pink', 'yellow', 'green'];
```

```
colors[1];
```

```
var colors = ['pink', 'yellow', 'green'];
```

```
colors[2];
```


```
var colors = ['pink', 'yellow', 'green'];
```

```
colors.length
```

FUNCTIES


DECLAREREN VAN EEN FUNCTIE

KEYWORD



```
function sayHello() {  
    document.write('Hello');  
}
```


FUNCTION NAME



```
function sayHello() {  
    document.write('Hello');  
}
```

```
function sayHello() {  
    document.write('Hello');  
}
```

CODE BLOCK (IN CURLY BRACES)

AANROEPEN VAN EEN FUNCTIE


```
sayHello();
```



FUNCTION NAME


DECLAREREN VAN
EEN FUNCTIE DIE
INFORMATIE
NODIG HEEFT

PARAMETER **PARAMETER**



```
function getArea (width, height) {  
    return width * height;  
}
```

```
function getArea(width, height) {  
    return width * height;  
}
```

The diagram consists of two horizontal brackets positioned below the words 'width' and 'height' in the return statement. Each bracket has a vertical line extending downwards from its center, pointing towards the text below.

**DE PARAMETERS WORDEN GEBRUIKT
ALS VARIABELEN BINNEN DE FUNCTIE**

AANROEPEN VAN
EEN FUNCTIE DIE
INFORMATIE
NODIG HEEFT


```
getArea (3, 5) ;
```



ARGUMENTS

LOCALE VS GLOBALE VARIABELEN

LOKAAL VS GLOBAL

```
function getArea(width, height) {  
    var area = width * height;  
    return area;  
}
```

```
var wallSize = getArea(3, 2);  
document.write(wallSize);
```

OBJECTEN

OBJECTEN: LITERAL NOTATION

NAMEN (KEYS)

```
var hotel = {  
  name: 'Quay',  
  rooms: 40,  
  booked: 25,  
  gym: true,  
  roomTypes: ['twin', 'double', 'suite'],  
  checkAvailability: function() {  
    return this.rooms - this.booked;  
  }  
};
```

WAARDEN

```
var hotel = {  
  name: 'Quay',  
  rooms: 40,  
  booked: 25,  
  gym: true,  
  roomTypes: ['twin', 'double', 'suite'],  
  checkAvailability: function() {  
    return this.rooms - this.booked;  
  }  
};
```

EIGENSCHAPPEN

```
var hotel = {  
  name: 'Quay',  
  rooms: 40,  
  booked: 25,  
  gym: true,  
  roomTypes: ['twin', 'double', 'suite'],  
  checkAvailability: function() {  
    return this.rooms - this.booked;  
  }  
};
```


METHODE

```
var hotel = {  
  name: 'Quay',  
  rooms: 40,  
  booked: 25,  
  gym: true,  
  roomTypes: ['twin', 'double', 'suite'],  
  checkAvailability: function() {  
    return this.rooms - this.booked;  
  }  
};
```

OBJECTEN: CONSTRUCTOR NOTATION

EIGENSCHAPPEN

```
function Hotel(name, rooms, booked) {  
    this.name = name;  
    this.rooms = rooms;  
    this.booked = booked;  
    this.checkAvailability = function() {  
        return this.rooms - this.booked;  
    };  
}
```

METHODEN

```
function Hotel(name, rooms, booked) {  
  this.name = name;  
  this.rooms = rooms;  
  this.booked = booked;  
  this.checkAvailability = function() {  
    return this.rooms - this.booked;  
  };  
}
```

AANMAKEN OBJECT


```
var quayHotel = new Hotel('Quay', 40, 25);
```

OBJECT

CONSTRUCTOR FUNCTIE

TOEGANG TOT OBJECTEN

```
var hotelName = hotel.name;
```



A horizontal line with vertical end caps is positioned below the word 'hotel'. A vertical line descends from the center of this horizontal line to the word 'OBJECT'.


OBJECT

```
var hotelName = hotel.name;
```



PROPERTY


```
var hotelName = hotel[ 'name' ];
```



OBJECT

A diagram consisting of a horizontal line with short vertical bars at each end, positioned below the word 'hotel'. A single vertical line extends downwards from the center of this horizontal line to the word 'OBJECT'.

```
var hotelName = hotel['name'];
```



PROPERTY

UPDATEN VAN OBJECTEN

```
hotel.name = 'Park';
```

```
hotel.name = 'Park';
```



OBJECT

```
hotel.name = 'Park';
```



PROPERTY

```
hotel.name = 'Park';
```

NIEUWE WAARDE

```
hotel[ 'name' ] = 'Park';
```



```
hotel[ 'name' ] = 'Park';
```



OBJECT

```
hotel[ 'name' ] = 'Park';
```



PROPERTY

```
hotel[ 'name' ] = 'Park';
```

NIEUWE WAARDE

INGEBOUWDE OBJECTEN

BROWSER OBJECT MODEL

EIGENSCHAPPEN:

`window.innerHeight`

`window.innerWidth`

`window.screenX`

`window.screenY`

METHODEN:

`window.print()`

DOCUMENT OBJECT MODEL

EIGENSCHAPPEN:

`document.title`

`document.lastModified`

METHODEN:

`document.write()`

`document.getElementById()`

GLOBAL JAVASCRIPT OBJECTS

EIGENSCHAPPEN:

```
saying.length
```

METHODEN:

```
saying.toUpperCase()
```

```
saying.toLowerCase()
```

```
saying.charAt(3)
```

CONTROLESTRUCTUREN

IF-STATEMENT

```
if (score > 50) {  
    document.write('You passed!');  
} else {  
    document.write('Try again...');  
}
```

VERGELIJKINGS- OPERATOREN

= =

IS GELIJK AAN

!=

IS NIET GELIJK AAN

= = =

STRICT GELIJK AAN

!==

STRICT NIET GELIJK AAN

>

GROTER DAN

<

KLEINER DAN

>=

GROTER OF GELIJK AAN

<=

KLEINER OF GELIJK AAN

LOGISCHE OPERATOREN

& &

LOGISCHE EN

||

LOGISCHE OF

!


LOGISCHE NIET

SWITCH STATEMENTS


```
switch (level) {  
  case 'One':  
    title = 'Level 1';  
    break;  
  
  case 'Two':  
    title = 'Level 2';  
    break;  
  
  case 'Three':  
    title = 'Level 3';  
    break;  
  
  default:  
    title = 'Test';  
    break;  
  
}
```

LOOPS

KEYWORD




```
for (var i=0; i<3; i++) {  
    document.write(i);  
}
```

CONDITION (COUNTER)

```
for (var i=0; i<3; i++) {  
    document.write(i);  
}
```

De variabele `i` wordt gedeclareerd
en krijgt als waarde 0

INITIALIZATION



```
for (var i=0; i<3; i++) {  
    document.write(i);  
}
```

Elke keer er een loop begint, wordt er nagegaan of i kleiner is dan 3

CONDITION

```
for (var i=0; i < 3; i++) {  
    document.write(i);  
}
```

Als `i` kleiner is dan 3,
dan wordt het code block uitgevoerd

```
for (var i=0; i<3; i++) {  
    document.write(i);  
}
```

De variabele `i` kan gebruikt worden binnen de loop

```
for (var i=0; i<3; i++) {  
    document.write(i);  
}
```

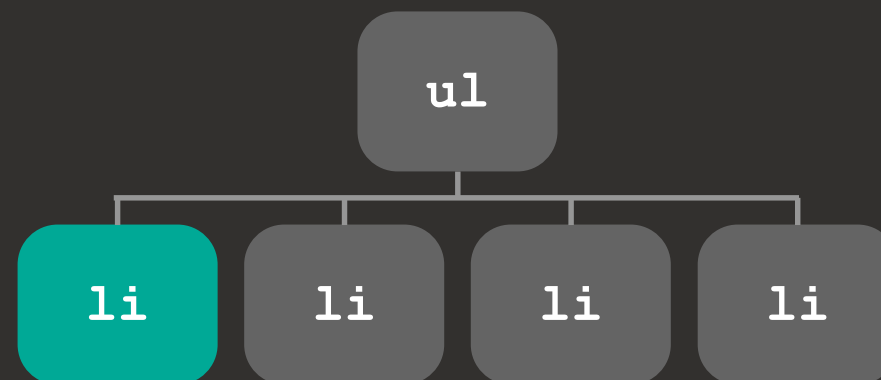

Wanneer de code binnen de accolades is uitgevoerd, zal de variabele `i` met 1 verhoogd worden.

```
                                UPDATE  
                                ┌───┴───┐  
for (var i=0; i<3; i++) {  
    document.write(i);  
}
```

ELEMENTEN SELECTEREN & UPDATEN

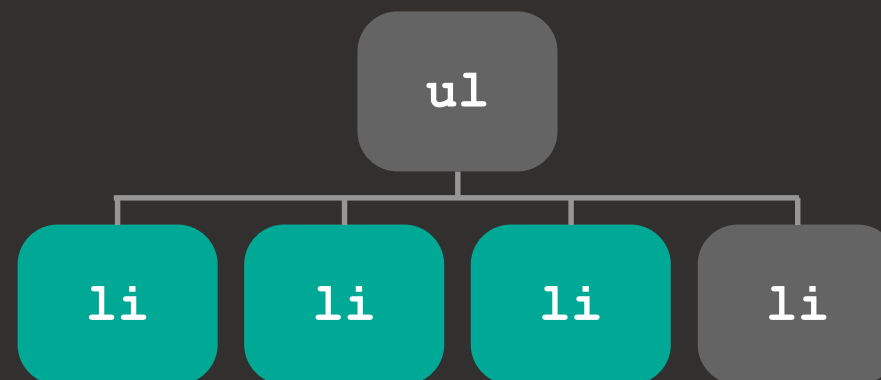
ELEMENTEN SELECTEREN

```
<ul>  
  <li id="one" class="hot">fresh figs</li>  
  <li id="two" class="hot">pine nuts</li>  
  <li id="three" class="hot">honey</li>  
  <li id="four">balsamic vinegar</li>  
</ul>
```



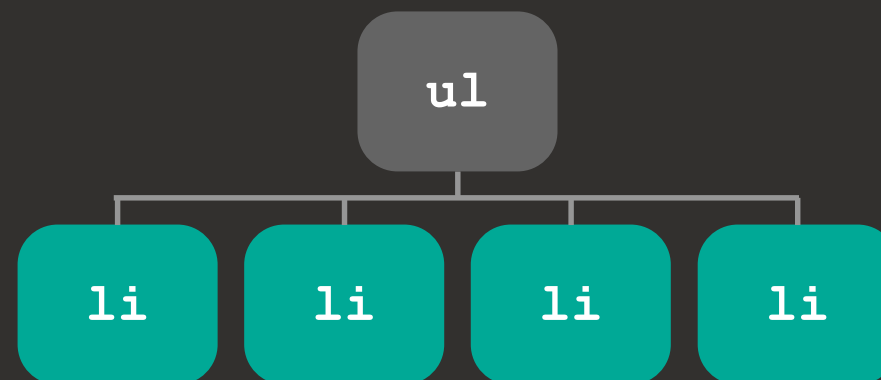
```
document.getElementById('one');
```

```
<ul>  
  <li id="one" class="hot">fresh figs</li>  
  <li id="two" class="hot">pine nuts</li>  
  <li id="three" class="hot">honey</li>  
  <li id="four">balsamic vinegar</li>  
</ul>
```



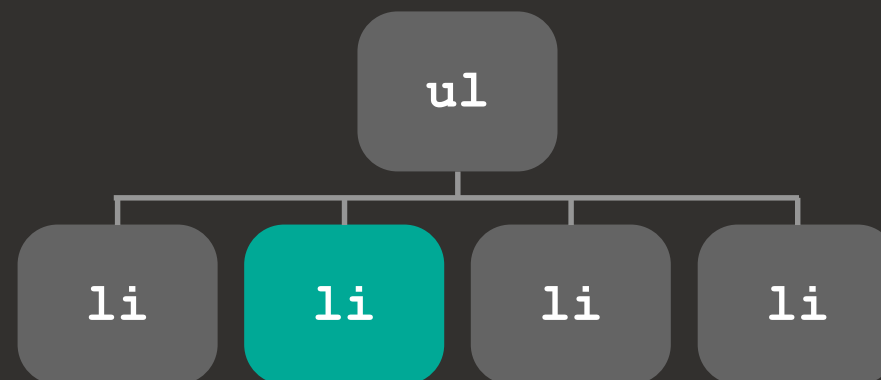
```
document.getElementsByClassName('hot');
```

```
<ul>  
  <li id="one" class="hot">fresh figs</li>  
  <li id="two" class="hot">pine nuts</li>  
  <li id="three" class="hot">honey</li>  
  <li id="four">balsamic vinegar</li>  
</ul>
```



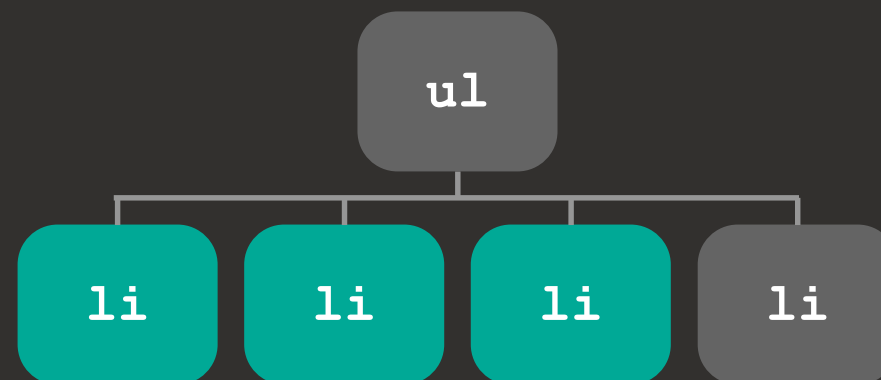
```
document.getElementsByTagName('li');
```

```
<ul>
  <li id="one" class="hot">fresh figs</li>
  <li id="two" class="hot">pine nuts</li>
  <li id="three" class="hot">honey</li>
  <li id="four">balsamic vinegar</li>
</ul>
```



```
document.querySelector('#two');
```

```
<ul>  
  <li id="one" class="hot">fresh figs</li>  
  <li id="two" class="hot">pine nuts</li>  
  <li id="three" class="hot">honey</li>  
  <li id="four">balsamic vinegar</li>  
</ul>
```



```
document.querySelectorAll('li.hot');
```


NODELISTS

Als een DOM query meerdere elementen terug geeft, is het bekend als een **NodeList**.

Items in een NodeList zijn genummerd en geselecteerd als een array:

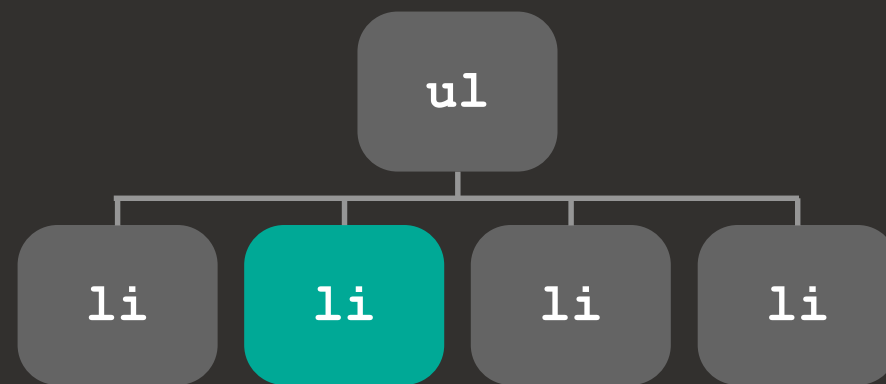
```
var elements;  
  
elements = document.getElementsByClassName('hot');  
  
var firstItem = elements[0];
```

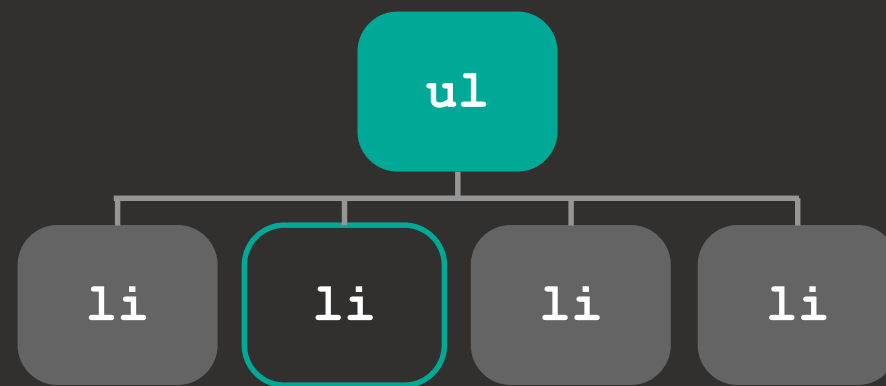
Je kan controleren of er elementen in de NodeList zitten voor je deze gebruikt:

```
if (elements.length >= 1) {  
    var firstItem = elements[0];  
}
```

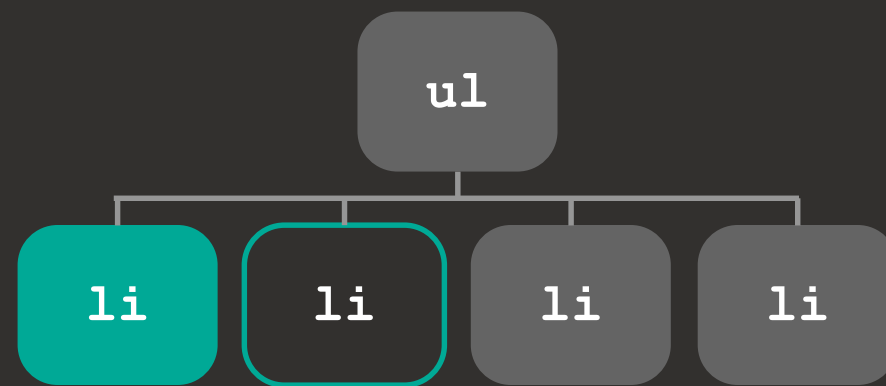
DOORKRUISEN VAN HET DOM

START ELEMENT

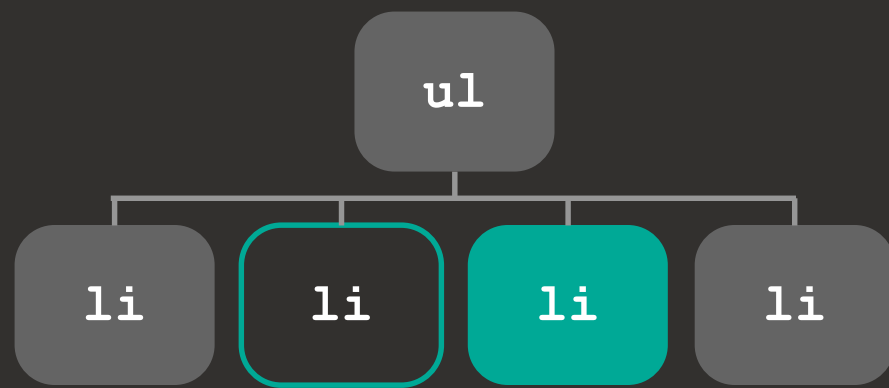




parentNode

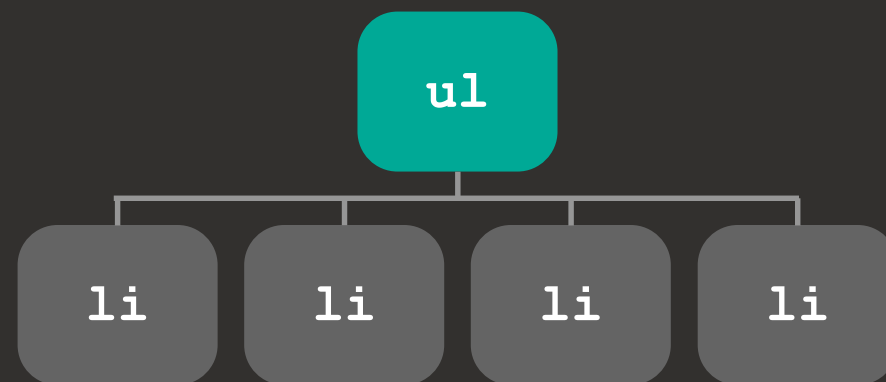


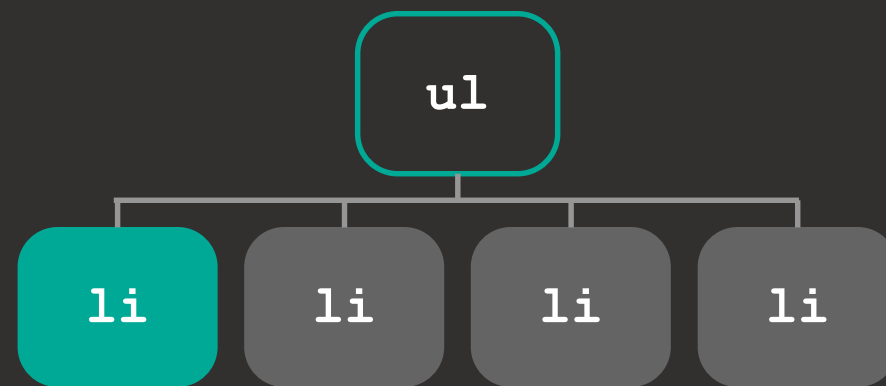
`previousSibling`



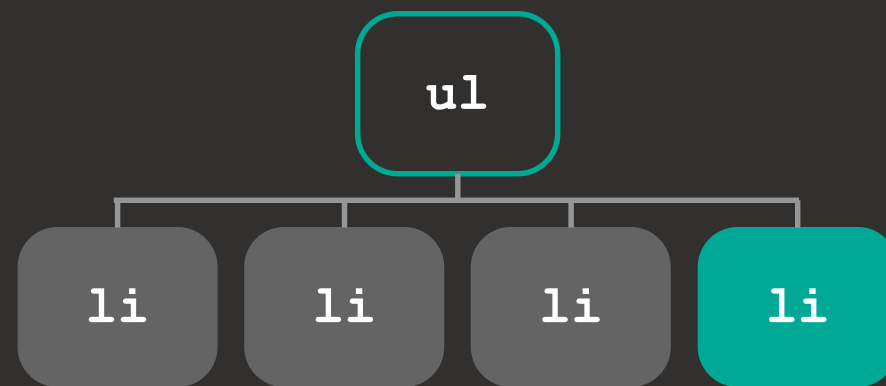
`nextSibling`

START ELEMENT





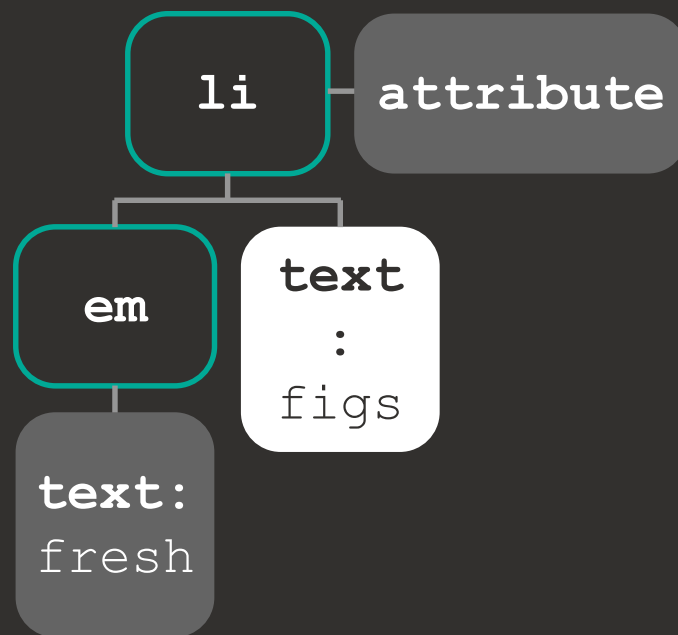
`firstChild`



`lastChild`

WERKEN MET ELEMENTEN

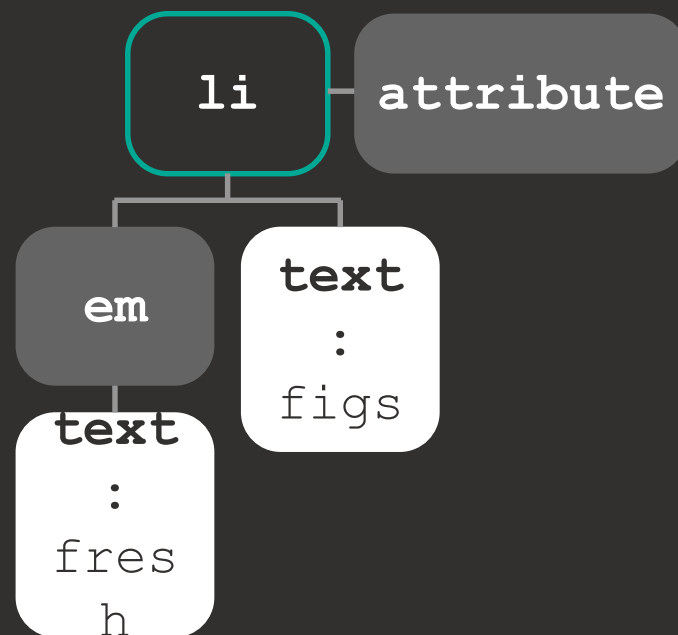
nodeValue werkt op text nodes



```
var el = document.getElementById('one');  
el.firstChild.nextSibling.nodeValue;
```

returns: figs

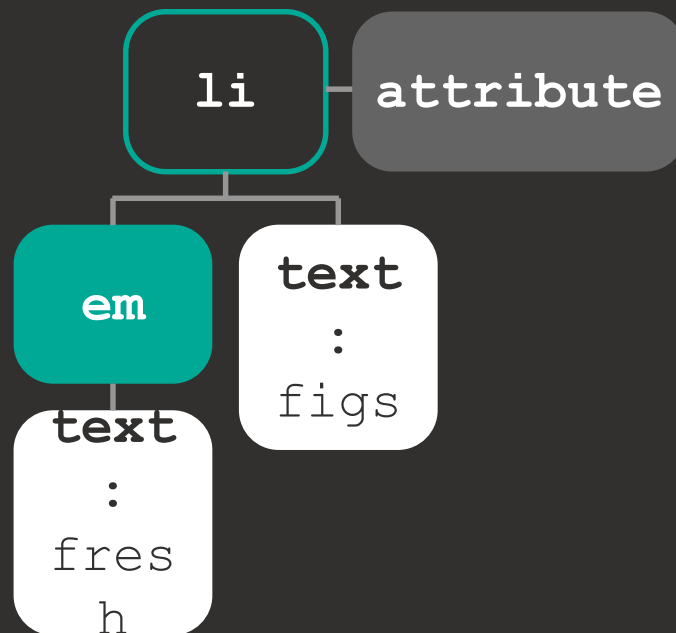
textContent verzamelt tekst inhoud



```
document.getElementById('one').textContent;
```

returns: fresh figs

innerHTML neemt de text en markup



```
document.getElementById('one').innerHTML;
```

returns: `fresh figs`

DOM MANIPULATIE

VS

innerHTML

```
createElement()  
createTextNode()  
appendChild()  
removeChild()
```

- Bouwt een string
- Bevat markup
- Update elementen

WERKEN MET ATTRIBUTEN

TOEGANG TOT EEN ATTRIBUUT

1. Een DOM query gebruiken om element te selecteren:

```
var el = document.getElementById('one');
```

2. Een methode haalt de attribuut op van een element:

```
el.getAttribute('class');
```

ATTRIBUUT UPDATEN

Controleer op attribuut en update;

```
var el = document.getElementById('one');  
  
if (el.hasAttribute('class')) {  
    el.setAttribute('class', 'cool');  
}
```

ATTRIBUUT VERWIJDEREN

Controleer op attribuut en verwijder;

```
var el = document.getElementById('one');  
  
if (el.hasAttribute('class')) {  
    el.removeAttribute('class');  
}
```

EVENT TOEVOEGEN

VERSCHILLENDE EVENT TYPES

USER INTERFACE EVENTS

load

unload

error

resize

scroll

KEYBOARD EVENTS

keydown

keyup

keypress

MOUSE EVENTS

click

dblclick

mousedown

mouseup

mouseover

mouseout

FOCUS EVENTS

focus / focusin
blur / focusout

FORM EVENTS

input

change

submit

cut

copy

paste

select

EEN EVENT KOPPELEN AAN EEN ELEMENT

HTML EVENT HANDLER ATTRIBUUT (NIET GEBRUIKEN)

```
<input type="text" id="username"  
  onblur="checkUsername()">
```



EVENT

HTML EVENT HANDLER ATTRIBUUT (NIET GEBRUIKEN)

```
<input type="text" id="username"  
      onblur="checkUsername()">
```



FUNCTION

TRADITIONAL DOM EVENT HANDLERS

```
e1.onblur = checkUsername;
```



ELEMENT

TRADITIONAL DOM EVENT HANDLERS

```
el.onblur = checkUsername;
```



EVENT

TRADITIONAL DOM EVENT HANDLERS

```
el.onblur = checkUsername;
```



FUNCTION

EVENT LISTENERS

```
el.addEventListener('blur', checkUsername, false);
```



ELEMENT

EVENT LISTENERS

```
el.addEventListener('blur', checkUsername, false);
```



EVENT

EVENT LISTENERS

```
el.addEventListener('blur', checkUsername, false);
```



FUNCTION

PARAMETERS MET EVENT LISTENERS

```
e1.addEventListener('blur', function() {  
    checkUsername(5);  
}, false);
```

Als 2e argument gebruiken
we een anonieme functie.