



WEB разработка

JavaScript

DOM манипулации

Съдържание

- DOM елементи
- Обхождане на DOM - дървото
 - Добавяне, премахване, променяне на елементи
- Промяна на DOM - дървото
- Оптимизиране на DOM операциите

DOM елементи

DOM елементи - преговор

Всеки **DOM елемент** е JavaScript обект, съответстващ на елемент от HTML.

- **Селектираме ги**, като използваме някой от DOM селекторите.
- **Създаваме ги** динамично с код.

DOM елементи

DOM елементите могат да бъдат променяни

Следните промени се прилагат веднага върху DOM и HTML страницата

```
//промяна съдържанието на div-елемент
```

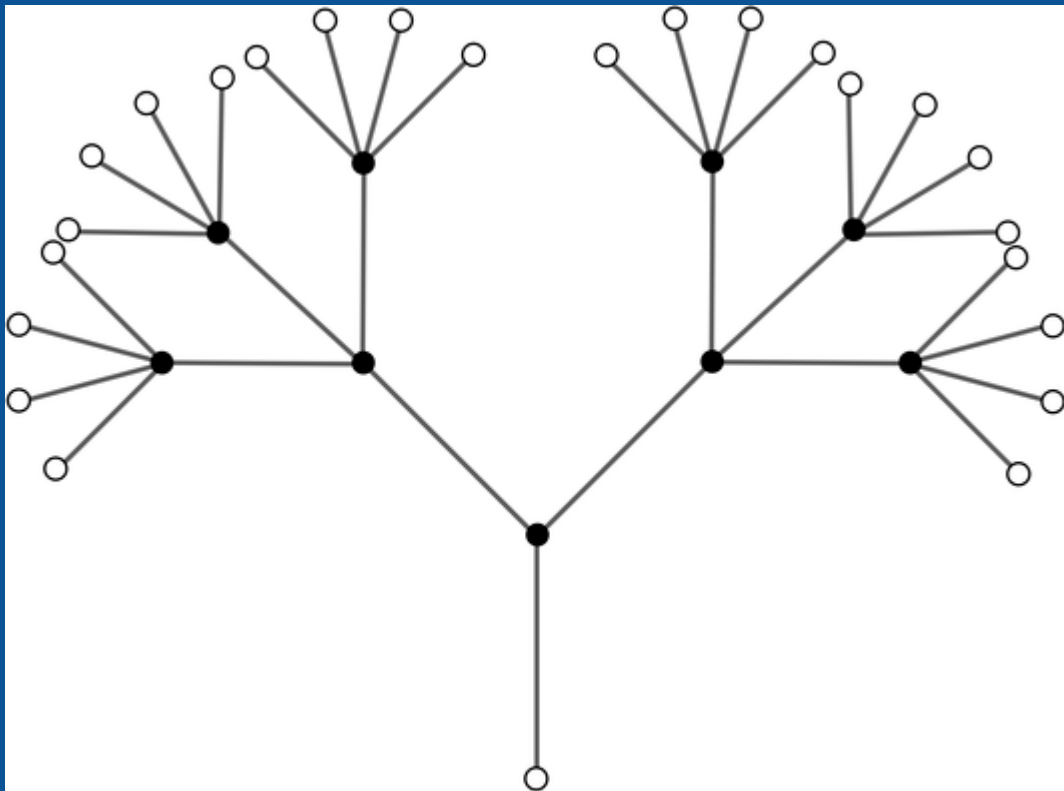
```
selectedDiv.innerHTML = "changed";
```

```
//променяме фона на div на "#456"
```

```
selectedDiv.style.background = "#456";
```

демо

Обхождане на DOM



Обхождане на DOM

- DOM елементите притежават свойства, отнасящи се до тяхната позиция в DOM - дървото:
 - Техния родител - **parent**
 - Техните деца - **children**
 - Техните роднини- **siblings**
 - Елементите непосредствено преди и след елемента.

Тези свойства могат да бъдат използвани за обхождане на DOM - дървото.

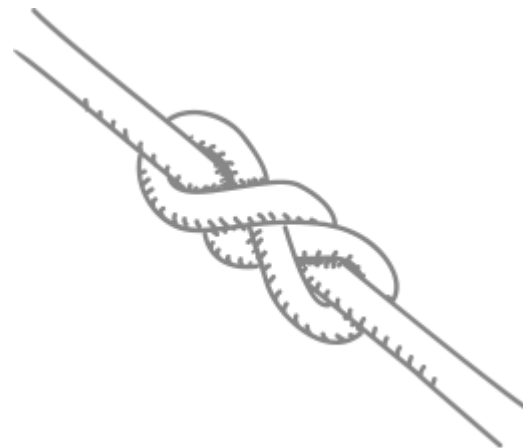
Обхождане на DOM

parentNode

- `element.parentNode`
 - Връща директния родител на елемента
 - Родителя на `document` е `null`

childNodes

- `element.childNodes`
 - Връща `nodeList` с всички деца - **child nodes**
 - Включително текста - **text nodes**
 - Включително whitespaces



Traversing the DOM

- DOM елементите имат някои свойства за специалните елементи, намиращи се около тях -
 - **first /last** child nodes
 - Елементите преди/след избран елемент/node

Свойства -

- firstChild / lastChild
- nextSibling / nextElementSibling
- previousSibling / previousElementSibling

Манипулиране на DOM - дървото

Манипулиране на DOM - дървото

Може да променяме DOM - дървото динамично, с помощта на JS

- Създаваме HTML елементи
- Премахваме HTML елементи
- Променяне на HTML елементите -
 - Тяхното съдържание
 - Стилизирането им
 - Техните атрибути - src, href

createElement()

DOM притежава метод за създаване на HTML елементи

- document.createElement(elementName)
- Връща обект от съответния тип HTML елемент.

```
//createElement.cloneNode(true)
```

```
var liElement = document.createElement("li");
```

appendChild()

- Когато създаваме HTML елемент динамично, той е просто един **JavaScript обект**, който
 - Все още **не е част от DOM** - дървото /на веб страницата/.
 - Новият HTML елемент трябва **да бъде добавен /appended/** към DOM - дървото.

```
var studentsList = document.createElement("ul");
    var studentLi = document.createElement("li");
        studentsList.appendChild(studentLi);
            document.body.appendChild(studentsList);
```

insertBefore()

- DOM API поддържа методи за вмъкване на елемент преди и след конкретен елемент.
 - `appendChild()` вмъква елемента винаги в края
 - `parent.insertBefore(newNode, specificElement)`

**Премахване
на
елементи**

removeChild()

- Премахваме елемент с `element.removeChild(eIToRemove)`
 - Подаваме елемента, който ще премахваме на неговия **родител**

**Променяне
на
елемент/и**

Променяне на елемент/и

- DOM елементите могат да бъдат премахнати и/или променени
 - Както конкретен елемент, така и неговите деца.
- DOM API дава възможност всеки елемент от DOM - дървото да бъде променян
 - Променяйки свойствата му/properties
 - Променяйки външния му вид

Променяне на елемент/и

Всеки HTML елемент е уникален в DOM - дървото

- Ако променим с JavaScript неговия външен вид или позиция - той си е все още същия елемент - обект.

**Променяне
на
Стила
на
елемент**

element.style.property

- Стилът на всеки HTML елемент, може да бъде променян с JavaScript
 - Като променяме стиловите атрибути
 - Променяме - inline - стила, не CSS документ

/Какъв е приоритетът на inline - стила, спрямо останалите начини за задаване на стил на HTML елементите?/

element.style.property

```
var div = document.getElementById("content");  
div.style.display = "block";  
div.style.width = "123px";
```

Оптимизиране на DOM операциите

Оптимизация на операциите с DOM

Добавянето на елементи към DOM - дървото е много бавна операция

- Когато добавяме елемент към DOM - дървото, то се зарежда отначало
- Препоръчително е новосъздадените елементи да бъдат добавяни заедно.

documentFragment

За целта използваме елемента **DocumentFragment**

- Той е малък **DOM** елемент, без родители
- Използва се за съхранение на готовите за прикачване, новосъздадени елементи едновременно към DOM - дървото.

documentFragment

- Прикачването на DocumentFragment - елемента към DOM - дървото, добавя само неговите деца - елементи.

```
var dFrag = document.createDocumentFragment();
    dFrag.appendChild(div);
    //appending more elements
.....
document.body.appendChild(dFrag);
```