

DOM入門

2009/2/12

WEBシステムグループ第八技術チーム

嶋崎 聖

アジェンダ

- DOMの基本
- DOMの構造
- DOMの利用

DOMの基本

- DOMって何だ？ -

DOMの基本

DOMとは . . .

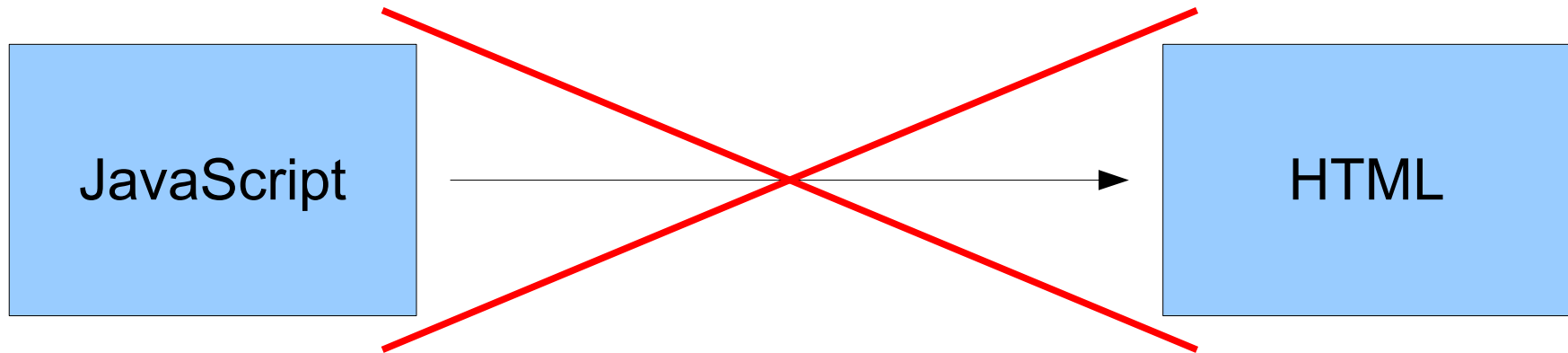
Document Object Modelの略で、

HTML文書やXML文書を

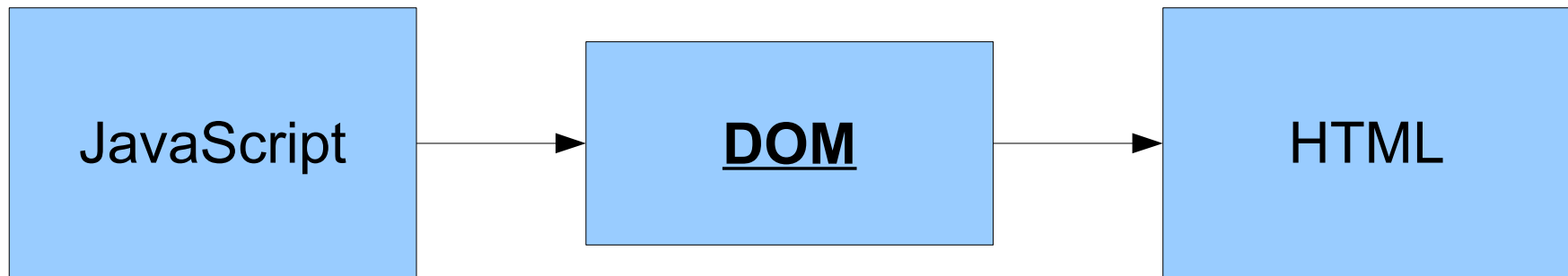
アプリケーションから利用するためのAPI

Wikipediaより

イメージ



- JavaScript単体では「HTML」データにアクセスできない



- DOMというAPIを通して「HTML」データにアクセスする

ただし、

DOM自体は単なる仕様

どう実装するかは

各ブラウザに依存

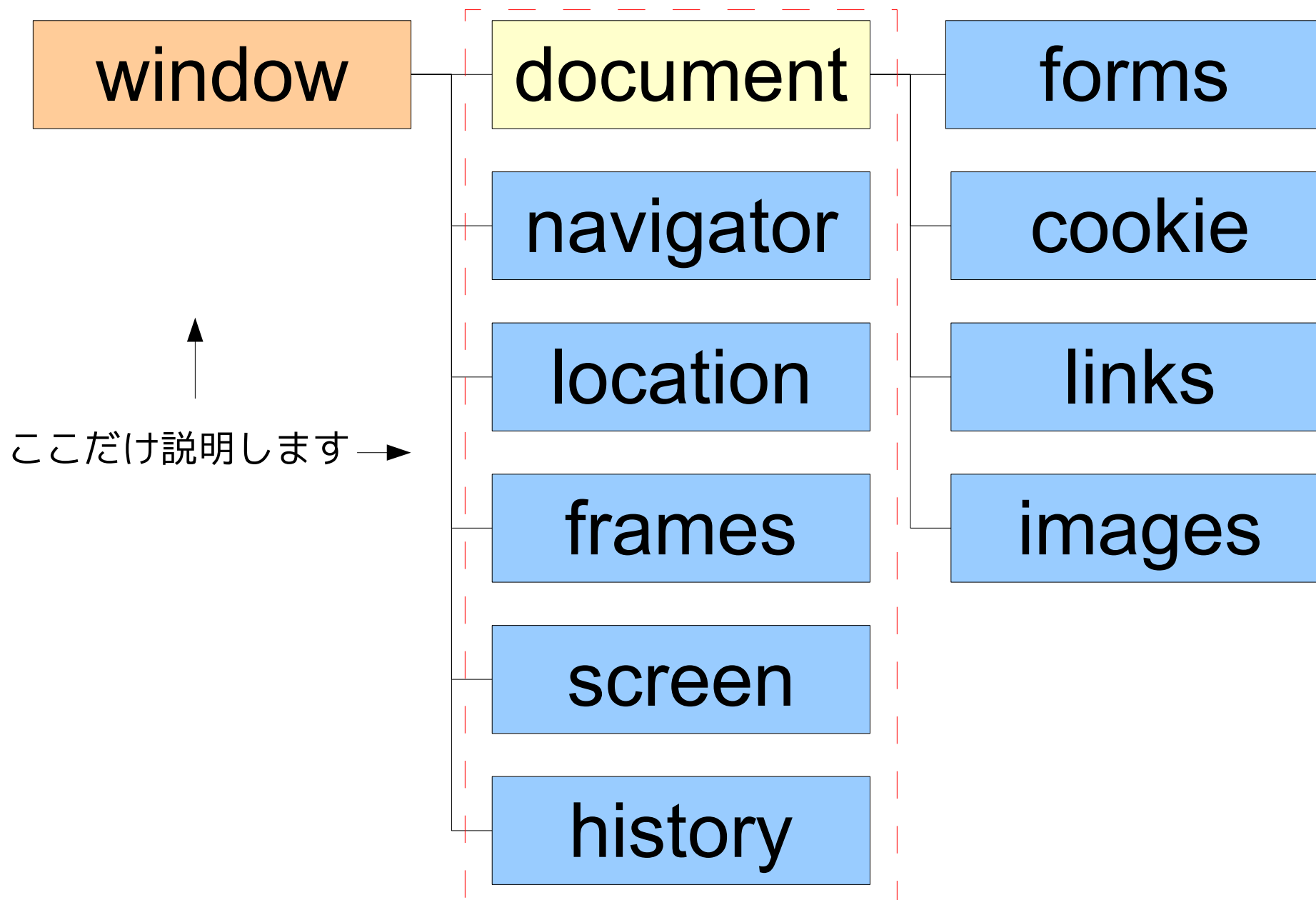


ブラウザごとに挙動が変わる

DOMの構造

– オブジェクト構成 –

DOMの構造



windowオブジェクト

- DOM全体を表すトップレベルオブジェクト
- ブラウザ全体をオブジェクトとして表したものの
- 省略することができる

例：

```
window.open( "http://www.mythos-jp.com");
```

```
alert( "hoge");// windowを省略している
```

```
//当然open()とも書けるし、window.alert()とも書ける。
```

navigator オブジェクト

- ブラウザそのものの情報を格納しているオブジェクト

例：

```
navigator.appName; //ブラウザの名称
```

```
// 参考IE7の場合：Microsoft Internet Explorer
```

locationオブジェクト

- ページのアドレスに関する情報を格納しているオブジェクト

例：

```
// 今のURLを表示
```

```
alert(location.href);
```

```
// mythosWEBサイトに移動
```

```
location.href = 'http://www.mythos-jp.com';
```

framesオブジェクト

- フレームに関する情報を格納しているオブジェクト
- 基本的にはそれぞれのwindowオブジェクトへの参照

例：

```
<frameset cols="50%,50%">  
  <frame name="frm_a" src="a.html" />  
  <frame name="frm" src="b.html" />  
</frameset>  
  
  // frm_aからfrm_bという名前のフレームを  
  // hoge.comに移動  
  
  parent.frm_b.location.href='hoge.com';
```

screenオブジェクト

- 画面に関する情報を格納しているオブジェクト

例：

```
// 画面の色数  
screen.colorDepth
```

historyオブジェクト

- 履歴に関する情報を格納しているオブジェクト

例：

```
// 履歴の一つ前に戻る  
history.back();
```

document オブジェクト

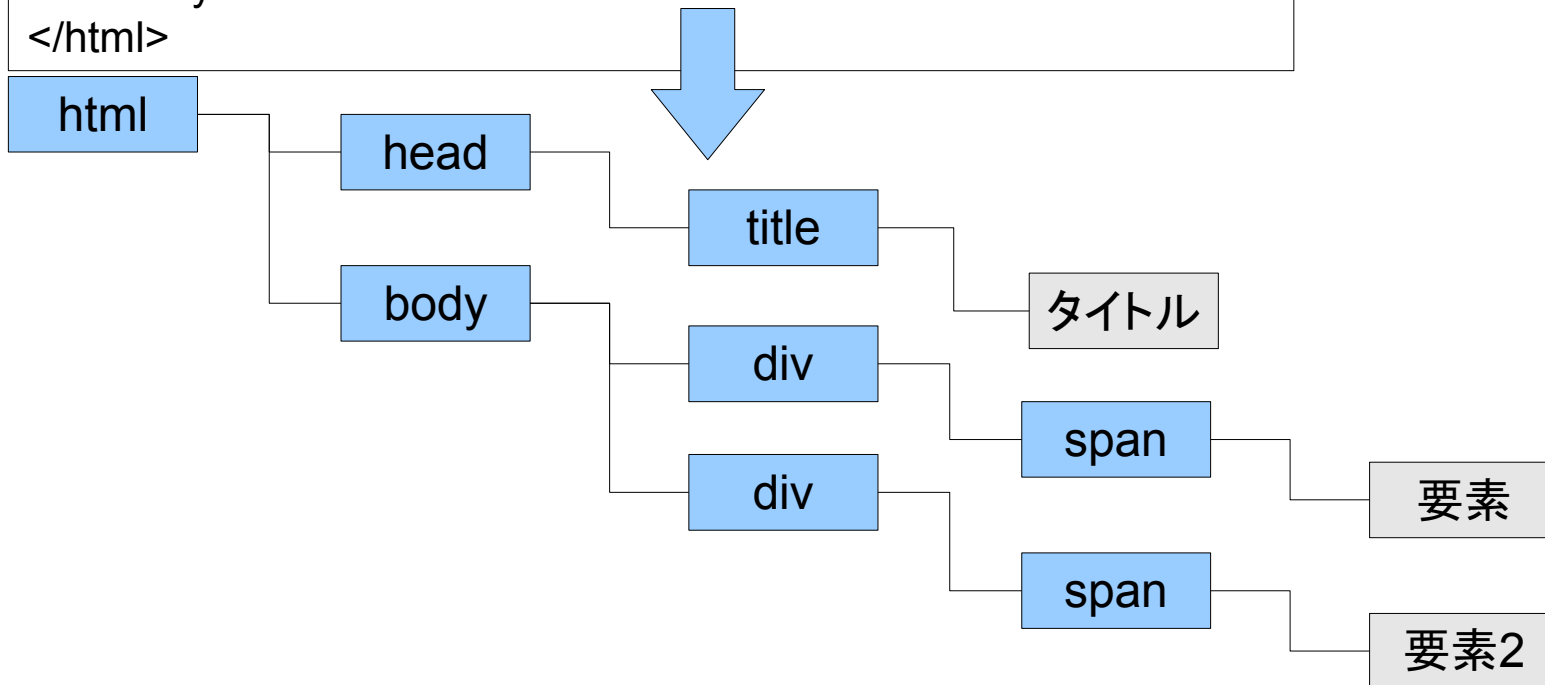
- ブラウザが描画している情報を格納しているオブジェクト
- Document Object

DOMの利用

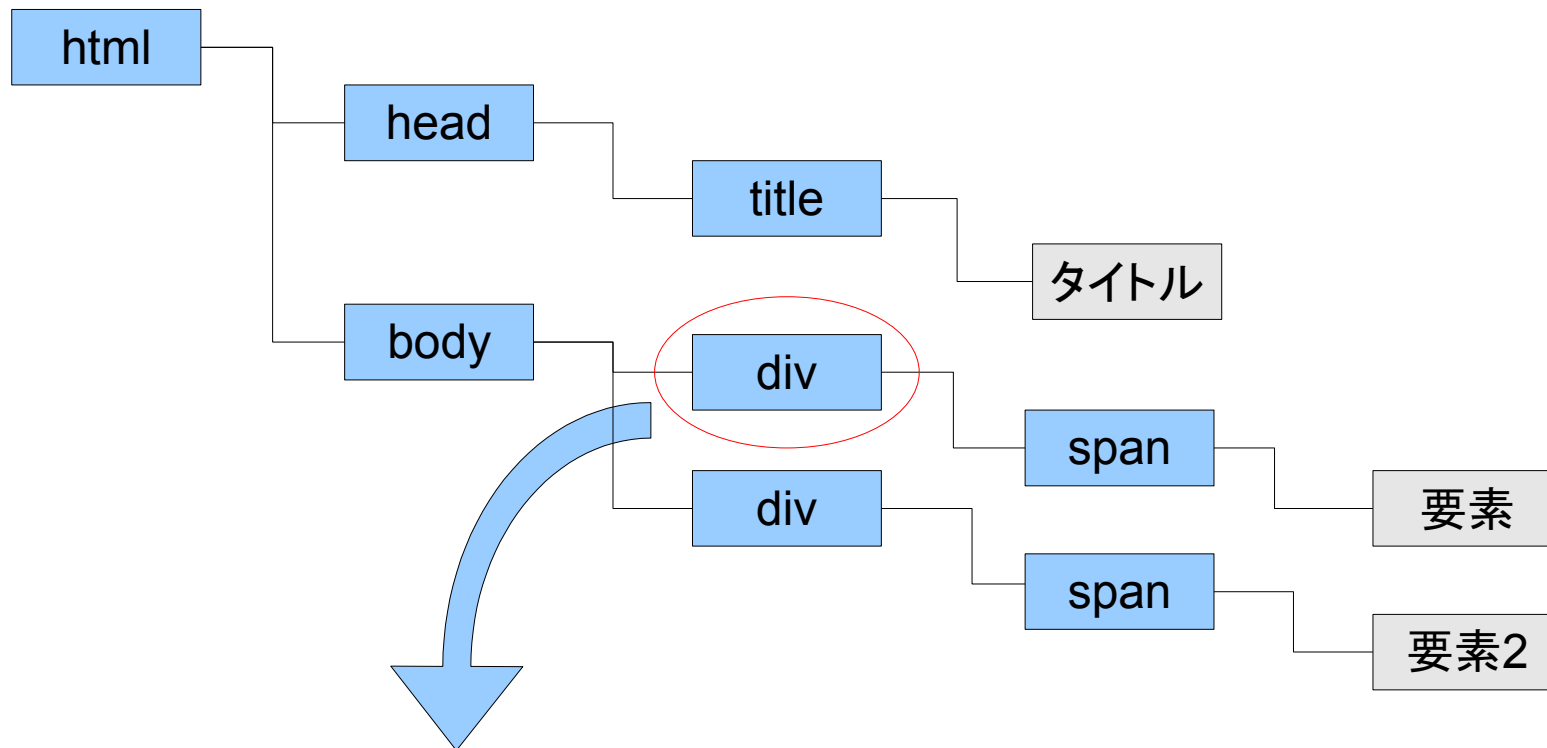
- どのようにデータにアクセスする? -

ノード (ドキュメントツリー)

```
<html>
  <head>
    <title>タイトル</title>
  </head>
  <body>
    <div id="main" align="center">
      <span>要素</span>
    </div>
    <div id="main2">
      <span>要素2</span>
    </div>
  </body>
</html>
```



エレメントの取得



element = document.getElementById("main")

返回值
エレメント

オブジェクト

getElementByIdメソッド

引数
(idの文字列)

エレメントの操作

```
element = document.getElementById("main")
```

返回值
エレメント

オブジェクト

getElementByIdメソッド

引数
(idの文字列)

- エレメントから各値にアクセス

```
alert( element.getAttribute('align') ); // center
```

```
alert( element.align ); // プロパティに直でもアクセスできる
```

//いくつかの属性はこのまま書き換えることも出来る

```
element.setAttribute('align','left');
```

```
element.align = 'left';
```

注意点その1

- class属性を扱うとき

```
element.getAttribute('class')
```

→Mozilla系統のみ有効

```
element.getAttribute('className');
```

→IEのみ有効

```
element.className;
```

→IE、Mozilla系統ともに有効

```
element.class;
```

→IE、Mozilla系統ともに無効

注意点その2

- 独自の属性を扱うとき

```
element.original
```

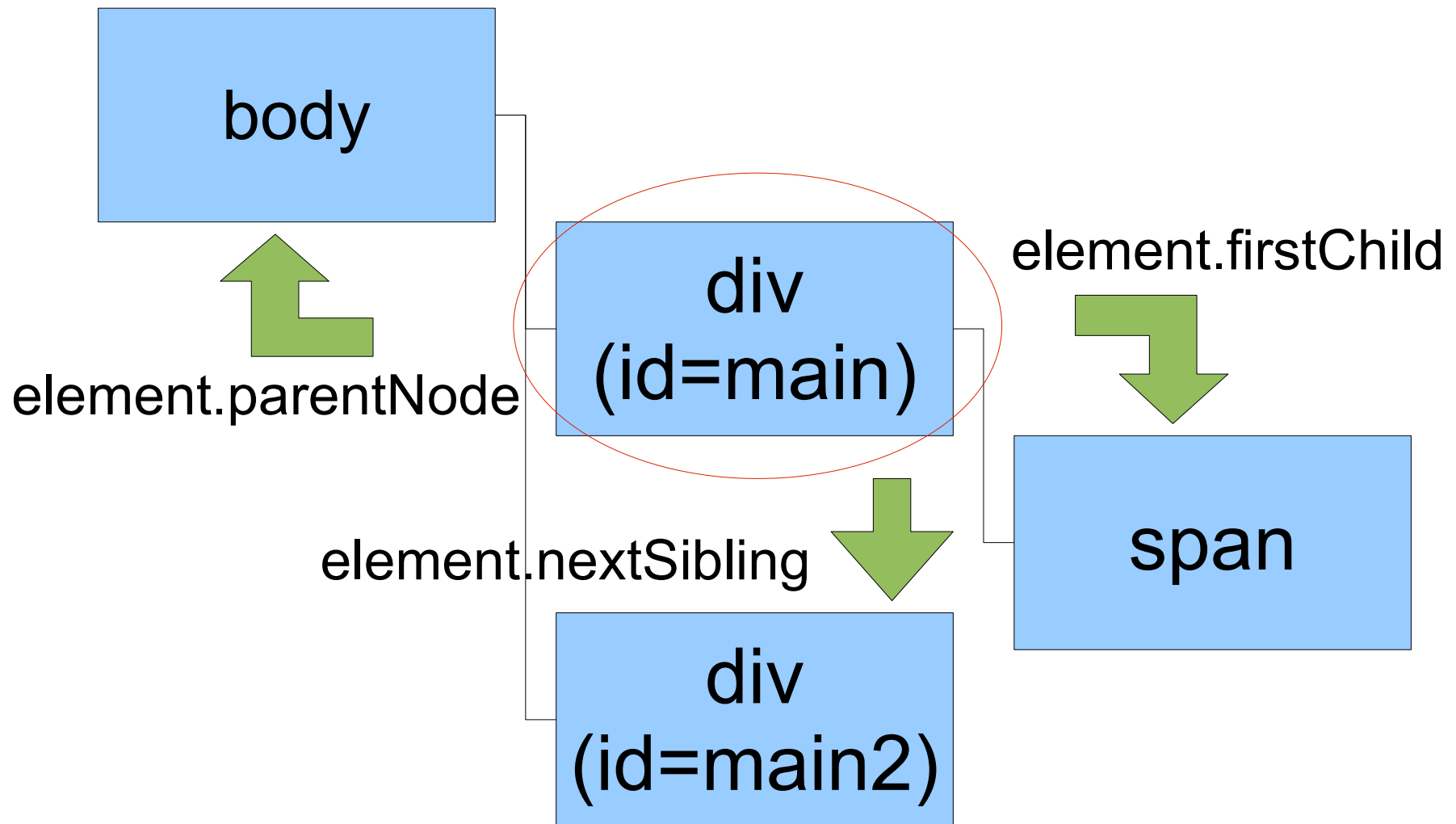
→IEのみ有効(ただし、推奨されない)

```
element.getAttribute('original');
```

→問題なし

エレメントからエレメントへのアクセス

- エレメントから各階層にアクセスできる



エレメントの操作まとめ

- parentNode
現在のノードの親のノード
- childNodes
子ノードのノードリスト
- firstChild
子ノードの中で最初のノード
- lastChild
子ノードの中で最後のノード
- previousSibling
一つ前のノード
- nextSibling
一つ後のノード

注意点

- document.getElementById(“main”).firstChildの指すエレメント

```
<div id="main">  
  <span>要素</span>  
</div>  
<div id="main2">  
  <span>要素2</span>  
</div>
```

IEの場合

Mozilla系統の場合

IEはエレメントの最初の改行を無視する。

Mozilla系統は最初の改行を無視しない。

その他のアクセス方法

- `getElementsByTagName()`

タグ名を引数に取り、ノードリストを返す

- `getElementsByName()`

`name`を引数に取り、ノードリストを返す

※ノードリストの実態は特殊なオブジェクト（≒配列）

こうじゃなくて、

```
<p id="hoge">  
  <span>hogeのSPAN</span><div>hogeのDIV</div>  
</p>  
<p id="hoge2">  
  <div>hoge2のDIV</div>  
</p>
```

```
var elem = document.getElementById( "hoge" );  
  
// hogeのSPAN  
alert( elem.firstChild.innerHTML );  
  
// hogeのDIV  
alert( elem.firstChild.nextSibling.innerHTML );
```

こう使う

```
<p id="hoge" >  
    <span>hogeのSPAN</span><div>hogeのDIV</div>  
</p>  
<p id="hoge2" >  
    <div>hoge2のDIV</div>  
</p>
```

```
var elem = document.getElementById( "hoge" );  
  
// id=hoge の中のdivタグのノードリストを取得  
var divList = elem.getElementsByTagName( "div" );  
  
// ノードリストの一つ目  
alert( divList[0].innerHTML ); // hogeのDIV
```

注意点その3

- name属性について

```
<input type="text" name="hoge" />
<a name="hoge">aaa</a>
<div name="hoge">bbb</div>
```

```
var nodeList = document.getElementsByName("hoge");
for(i=0;i<nodeList.length;i++){
    alert(nodeList[i].nodeName);
}
```

IEの場合 :

INPUT , A →name属性をサポートしないタグは無視される

Mozilla系統の場合 :

INPUT , A , DIV→name属性の全てを取得する

エレメントの作成

- 作成

```
element = document.createElement("a")
```


戻り値
エレメント

オブジェクト

createElementメソッド

引数
(タグ名)

- 設定

```
element.href = 'http://www.mythos-jp.com';
```

```
element.innerHTML = 'リンク';
```

- 追加

```
document.body.appendChild(element);
```

エレメントの作成まとめ

- `element = createElement(tagname)`
tagnameに指定された要素を作成
- `text = createTextNode(data)`
テキストを作成
- `element.appendChild(newelement)`
elementの子ノードの最後にnewelementを追加
- `element.insertBefore(newelement,refelement)`
elementの子ノードのrefelementの前にnewelementを追加
- `element.replaceChild(newelement,replaceelement)`
elementの子ノードのreplaceelementをnewelementで書き換え
- `element.removeChild(removeelement)`
elementの子ノードのremoveelementを削除

質問とかありますか？
