\* 브라우저 렌더링 과정 요약

브라우저는 우선 HTML, StyleSheets를 해석 -> HTML의 문서 해석이 끝나면 ‘DOM 트리 생성’ , StyleSheets 문서의 해석이 끝나면 ‘스타일 규칙’을 생성

렌더 트리 = ‘DOM트리’ + ‘스타일 규칙’

1. Reflow

생성된 DOM 노드의 레이아웃 수치(너비, 높이, 위치 등) 변경 시 영향 받은 모든 노드의(자신, 자식, 부모, 조상(결국 모든 노드)) 수치를 다시 계산하여, 렌더 트리를 재생성하는 과정

\*Reflow가 발생하는 대표적인 예)

-페이지 초기 렌더링 시(최초 Layout 과정)

-윈도우 리사이징 시(Viewport 크기 변경시)

-노드 추가 또는 제거

-요소의 위치, 크기 변경(left, top, margin, padding, border, width, height, 등..)

-폰트 변경과(텍스트 내용) 이미지 크기 변경(크기가 다른 이미지로 변경 시)

1. Repaint

-Reflow 과정이 끝난 후 재생성된 렌더 트리를 다시 그리게 되는 과정

-background-color, visibility, outline 등의 스타일 변경 시에는 레이아웃 수치가 변경되지 않으므로 Reflow 과정이 생략된 Repaint 과정만 일어나게 된다.

* 1. Reflow, Repaint 최소화

// bad

const body = document.body;

body.style.width = '50px';

body.style.height = '100px';

// good

const body = document.body;

body.style.cssText = 'width: 50px; heigh: 100px;'

요소에 스타일을 줄때는 cssText로 reflow가 한번만 실행되도록 한다.

3-2

// bad

const ulElement = document.getElementsByTagName('ul')[0];

for(let i=0; i<10; i++) {

  ulElement.innerHTML += `<li> list${i} </li>`;

}

// good

const ulElement = document.getElementsByTagName('ul')[0];

let strHtml = ulElement.innerHTML;

for(let i=0; i<10; i++) {

  strHtml += `<li> list${i} </li>`;

}

ulElement.innerHTML = strHtml;

여러 개의 엘리먼트가 추가된다면 위와 같이 가상의 형태로 추가할 엘리먼트를 구상하고 한번에 수정해서 reflow를 줄인다.

-그 외 position 속성을 ‘fixed’ or ‘absoulte’로 값을 주면 전체 노드에 걸쳐 Reflow비용이 들지 않는다.

-사용하지 않는 노드에는 visibility: invisible 보다 display:none 사용하기 (visibility의 경우 레이아웃 공간을 차지하기 때문에 reflow의 대상이 된다.)

-테이블의 경우 조금만 변화가 일어나도 테이블 전체가 reflow 되기 때문에 테이블 레이아웃 사용을 자제해야 한다.

Repaint만 일어나는 속성들

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Background | Background-image | Background-position | Background-size | Background-repeat |
| Border-radius | Border-style | Box-shadow | Color | Line-style |
| Outline | Outline-color | Outline-style | Outline-width | Text-decoration |
| visibility |  |  |  |  |