[WEB UI Study]

WEB Animation 1탄

NHN NEXT 우재우

반갑습니다~ 오랜만이에요 :)

기말고사 + 메르스 방학 덕분에

우리는 Javascript를 다 까먹었죠 ㅋ

지금까지 우리는 Javascript를 공부하면서

DOM을 선택하고 조작하고

이벤트를 발생시키는 법을 배웠어요!

오늘은 WEB Animation을 배워봅시다!

사실 저는 Animation 전문가라능!!!

WEB에서 Animation을 구현하는 방법은

크게 두 가지 방법이 있습니다

하나는 CSS Animation,

다른 하나는 JS Animation

그리고 JS Animation에는

또 두 가지 방법이 있어요

하나는 setTimeout, setInterval이 있고,

requestAnimationFrame이 있습니다.

오늘은 JS Animation만 공부할게요~

자, 하나씩 살펴봅시다!

참고로 오늘 만드는 모든 실습은

파란색 박스가 오른쪽으로 계속 가는

애니메이션 만들기입니다~

아래와 같은 HTML 코드를 준비해주세요.

```
<body>
    <div id="box" style="position: relative; top: 50px; left: 10px;
width: 50px; height: 50px; background-color: blue;"></div>
    <script src="test.js"></script>
</body>
```

참고로 Javascript에서

element.style.(속성)을 쓰려면

external css 파일이 아니라

inline으로 style에 추가되어 있어야 해요!

아래와 같은 HTML 코드를 준비해주세요.

```
<body>
    <div id="box" style="position: relative; top: 50px; left: 10px;
width: 50px; height: 50px; background-color: blue;"></div>
    <script src="test.js"></script>
</body>
```

JS Animation

setTimeout & setInterval

setTimeout(callback, milliseconds)

지정한 시간 이후에 callback함수를 실행해줍니다.

크롬 개발자도구 콘솔에 아래와 같이 입력해봅시다.

```
setTimeout(function(){ alert("Hello"); }, 3000);
```

setInterval(callback, milliseconds)

setInterval은 setTimeout이 계속 반복된다고 보시면 됩니다. 그래서 일정시간마다 callback에 등록한 함수를 계속 실행해주죠!

크롬 개발자도구 콘솔에 아래와 같이 입력해봅시다.

```
setInterval(function(){ alert("Hello"); }, 3000);
```

clearTimeout(timeoutID), clearInterval(intervalID)

setTimeout과 setInterval을 멈출 때는 clearTimeout과 clearInterval을 사용합니다. setTimeout과 setInterval의 리턴 값은 각각 timeout의 ID와 interval의 ID를 나타냅니다. clear의 인자로 넣어주면 멈춥니다!!

```
var si = setInterval(function(){ alert("World"); }, 3000);
clearInterval(si);
```

setTimeout으로 Animation 만들기

setTimeout은 딱 한 번만 실행되기 때문에 계속해서 실행하려면 재귀함수로 만들어줍니다.

예시)

```
function goRight() {
    box.style.left = parseInt(box.style.left) + 5 + 'px';
    setTimeout(goRight, 20);
}
setTimeout(goRight, 20);
```

setInterval로 Animation 만들기

setInterval은 계속 호출되기 때문에 굳이 재귀로 만들 필요는 없어요~

예시)

```
function goRight() {
    box.style.left = parseInt(box.style.left) + 5 + ‘px';
}
setInterval(goRight, 20);
```

set 시리즈의 단점

브라우저에서 Javascript는 싱글 뜨레드라고 누누히 얘기했습니다.
그래서 다른 Javascript 로직처리가 길어지다 보면 setTimeout과 setInterval 에서 지정해준 시간이 정확하게 작동하지 않을 수도 있습니다.

마치 밥 먹고 있는데 계속 누가 말을 시키면 밥을 제대로 먹지 못 하는 것과 같은 이치죠. 잉? ㅋㅋㅋㅋ

그뿐만이 아닙니다! 디스플레이가 갱신되는 주기와 맞지 않을 경우 불필요한 렌더링 또는 오작동 애니메이션이 발생하기도 하는데요, 보통 주파수가 60Hz 인 대부분의 디스플레이 갱신 주기는 16.67밀리초입니다. 만약 setInterval을 16.67보다 작게 주면 한 갱신 주기 안에 두 번의 애니메이션이 발생하겠죠?

정리하면...

set시리즈는 비효율적이다.

-> 애니메이션이 과도하게 그려진다.

-> CPU가 낭비된다. -> 추가 전력을 사용한다.

-> 에너지가 부족해진다.

-> 에너지를 더 생산해내기 위해 자원을 소모한다.

-> 자원을 더 채취하기 위해 환경을 파괴한다.

-> 지구멸망

결론적으로 말씀드리면

JS Animation에서

setTimeout & setInterval은

지구를 위해서 사용하지 않는 것이 좋습니다!

그래서 내가 나왔다!!!

requestAnimationFrame!!!

requestAnimationFrame(callback)

requestAnimationFrame이 앞서 살펴본 set 시리즈와 다른 점은 바로 개발자가 시간을 지정하지 않는다는 점입니다!! 대신 **바로 다음 렌더링 프레임 타이밍에 맞춰** callback함수를 실행할 뿐이죠. 그래서 주기가 일정한 렌더링 타이밍에 함수를 실행하기 때문에 불필요하거나 잘못된 애니메이션을 발생시킬 걱정이 없습니다.

requestAnimationFrame 역시 단일 callback만을 인자로 가지기 때문에 set 시리즈로 작성한 애니메이션도 쉽게 requestAnimationFrame으로 바꿀 수 있어요!

```
requestAnimationFrame( function() {alert( ‘ Hello!')} );
```

뭐라고 하는지 모르겠다.
그냥 존나 가만있어야겠다.

모르겠으면 JS Animation은
닥치고 requestAnimationFrame!

requestAnimationFrame으로 Animation 만들기

requestAnimationFrame으로 Animation을 만들 때는 setTimeout과 같은 방식으로 만들면 됩니다. 단지 시간만 지정해 주지 않았을 뿐이죠.

```
function goRightWithRAF (timestamp) {
    box.style.left = parseInt(box.style.left) + 5 + 'px';
    requestAnimationFrame(goRightWithRAF);
}
requestAnimationFrame(goRightWithRAF);
```

timestamp?!?!

requestAnimationFrame 예제를 찾다보면 현재 시간을 저장하기 위해서 경우에 따라 Date.now()로 현재 시간을 받아오는데, 그럴 필요가 전혀 없죠!

requestAnimationFrame은 **기본적으로 callback함수의 인자로 timestamp가 존재**합니다. timestamp는 해당 callback함수가 호출된 시점의 시간을 말합니다. 시간을 알 수 있으니 시간에 따른 변화를 코딩할 수 있게 되었어요. 물리학 시간에 배운 운동 방정식을 구현할 수 있다는!!!

다음 장은 3초 동안 상자를 800px까지 옮기고 멈추는 코드입니다. 차근차근 읽으면서 timestamp를 이용해서 어떻게 시간에 따른 변화를 반영하는지 살펴봅시다.

```
var startTime;
var startLeft = parseInt(box.style.left);

var unitDist = (800 - startLeft) / 3000;


function tick(timestamp) {
     if (!startTime) startTime = timestamp;
     var deltaTime = timestamp - startTime;

     box.style.left = Math.floor(Math.min(startLeft
                     + unitDist * deltaTime, 800)) + ‘px’;


     if (parseInt(box.style.left) < 800) {
          requestAnimationFrame(tick);
     }
}

requestAnimationFrame(tick);
```

```
// 초기값들을 저장해줍니다.
var startTime;
var startLeft = parseInt(box.style.left);
// 이동할 거리를 3000ms(3초)로 나누면 1ms 동안 이동하는 거리가 나오겠죠?
var unitDist = (800 - startLeft) / 3000;

// 프레임마다 호출될 tick함수입니다. timestamp를 기본 인자로 가지고 있어요!!!
function tick(timestamp) {
    if (!startTime) startTime = timestamp;
    var deltaTime = timestamp - startTime;

    // 물리시간에 배운 (거리) = (시간) * (속력)에 의해
    // unitDist * deltaTime은 애니메이션 시작부터 변화된 거리가 되겠죠?
    box.style.left = Math.floor(Math.min(startLeft
                    + unitDist * deltaTime, 800)) + 'px';

    // left가 800보다 작으면 계속 tick이 호출될거예요.
    if (parseInt(box.style.left) < 800) {
        requestAnimationFrame(tick);
    }
}
// 애니메이션을 시작하는 코드입니다! 이거 없으면 실행 안 되겠죠?
requestAnimationFrame(tick);
```

이벤트와 requestAnimationFrame를

적절히, 잘, 멋지게 이용하면

재미있는 애니메이션을 만들 수 있겠죠?

예를 들어서 이동 버튼을 누르면 상자가 움직이다가,

정지 버튼을 누르면 상자를 멈춘다거나...

강려크한 라이브러리 사용!

우리가 자주 쓰는 jQuery에도 .animate() 함수가 있고,

이외에도 TweenJS, Jsanim, animo.js, Move.js, Collie, GSAP

(GreenSock Animation Platform) 등 애니메이션을 지원하는

라이브러리가 많습니다.

그렇지만 언제나 말씀드리지만

라이브러리 너무 좋아하면 게을러져서 **소**가 될 수 있으니 주의합시다.

나중에 라이브러리 쓰는 법도 따로 공부를 해보죠~

공부할게 정말 드럽게 많네요 ^^

사실 지금 여러분께는 무지 어려울 거라 생각해요

저도 처음 애니메이션 배울 땐 뭔 소린지 몰랐거든요

일단 따라서 코딩해보고, 패턴을 차차 익히면

언젠가 애니메이션에 자신감이 생길거예요!!

다음 시간에는 CSS Animation을 공부합시다

:)