# 3주차 과제1 (시계) 기능 명세

|  |  |  |
| --- | --- | --- |
| 기능 번호 | 세부 기능 설명 | 추가기능여부 |
| FR1 | 화면에 배터리가 표시되며, 처음에 동작시키면 100%이고, 1초에 1%씩 감소합니다. | N |
| FR2 | 배터리가 0%가 되면, 시계 화면에서 시간이 표시되는 부분만  검정색 배경으로 안 보이게 해줍니다. | N |
| FR3 | 시/분/초를 설정하고 추가를 누르면 알람이 추가됩니다. | N |
| FR4 | 알람 현황이 업데이트됩니다. **최대 3개 알람**까지 가능합니다. | N |
| FR5 | 전원 버튼을 클릭 시 배터리가 재충전 되며 기능이 재활성화 됩니다. | Y |
| FR6 | 설정된 알람을 제거할 수 있습니다. | Y |

# 구현

## 구현 사항

|  |  |  |
| --- | --- | --- |
| 기능 번호 | 완료여부  (O,X) | 세부사항 |
| FR1 | O | <진행사항>  배터리를 다섯 칸으로 나누어 20초마다 하나씩 감소되게 구현 |
| FR2 | X | <진행사항>  배터리가 0%가 되면 시간을 표시하는 화면과 알람을 설정하는 화면이 검은색 배경이 됩니다.  <안된부분>  배터리가 0%가 되어도10초가 더 지나야 꺼집니다.  <상세 내용>  20 초마다 배터리가 1칸씩 감소하는데 0%가 되었을 때 10초가 더 감소해야 화면이 꺼집니다. |
| FR3 | O | <진행사항>  (+) 버튼을 클릭 후 (H/M/S) 버튼을 통해 각각 시/분/초를 설정할 수 있습니다. (1씩 증가)  이때 설정 중인 알람은 깜빡입니다.  시간 설정 후 (+) 버튼을 한 번 더 클릭하면 알람이 설정됩니다. |
| FR4 | O | <진행사항>  알람을 최대 3개까지 설정할 수 있습니다.  새로운 알람을 추가할 시 가장 처음에 설정해둔 알람이 삭제 됩니다. |
| FR5 | O | <진행사항>  전원 버튼은 평상 시 초록색 빛을 내다가 배터리가 다 될 시 빨간색 빛을 냅니다. 전원 버튼 클릭 시 배터리가 충전되며 안보였던 화면들도 다시 보입니다. |
| FR6 | O | <진행사항>  (-) 버튼 클릭 시 가장 마지막에 설정해둔 알람부터 삭제가 됩니다. |

## 웹 화면 구성

제목 : 시계 화면 (기본)

|  |  |
| --- | --- |
| 화면 | |
| 배지 단색으로 채워진배지 1 단색으로 채워진 | |
| 설명 | ① 날짜는 “년도.월.일 요일” 형식으로 표기되며 시간은 디지털 숫자로 “시:분:초” 형식으로 표기됩니다.  매 초 마다 시간이 갱신됩니다.  ② 배터리는 5칸으로 구성되어 있으며 배터리가 있는 동안에는  배터리 옆에 있는 전원 버튼이 초록빛을 냅니다.  20초 마다 배터리가 1칸씩 감소합니다. |

제목 : 알람 설정 시 시계 화면

|  |  |
| --- | --- |
| 화면 | |
| 배지 단색으로 채워진배지 1 단색으로 채워진 | |
| 설명 | ① 알람을 설정할 수 있는 버튼입니다. (+) 버튼을 통해 최초 등록을, (H/M/S)버튼을 통해 시간 설정을 할 수 있습니다. (설정 중인 알람은 깜빡입니다)  다시 한번 (+) 버튼 클릭 시 알람이 등록됩니다.  (–) 버튼을 통해 등록된 알람을 제거할 수 있습니다.  ② 1번 버튼들을 통해 설정된 버튼을 확인할 수 있는 화면입니다. |

제목 : 배터리 0%일 때 시계 화면

|  |  |
| --- | --- |
| 화면 | |
| 배지 3 단색으로 채워진배지 단색으로 채워진배지 1 단색으로 채워진 | |
| 설명 | ① 배터리가 0%가 될 시 전원 버튼이 빨간빛이 됩니다  전원 버튼 클릭 시 다시 초록 빛이 되며 배터리가 다시 5칸이 됩니다.  ②, 배터리가 0%가 될 시 화면이 검은색 배경이 됩니다.  배터리 회복 시 다시 화면이 표기가 됩니다. |

## 구현 내용

### HTML

|  |  |
| --- | --- |
| 파일명 | 구현 내용 |
| index.html | 시계 화면을 구성하고 css와 js 파일을 적용 합니다. |

### CSS

|  |  |
| --- | --- |
| 파일명 | 구현 내용 |
| reset.css | 브라우저의 기본 스타일을 초기화합니다. |
| global.css | 공통적으로 적용되는 스타일을 적용합니다. |
| \_mixin.scss | 호출해서 사용할 mixin 스타일들을 정의합니다. |
| style.scss | 화면에 적용할 세부 스타일을 정의합니다. |
| style.css | scss 파일을 css로 전환하여 화면에 스타일을 적용합니다. |

### JavaScript

파일 명 : battery.js

|  |  |
| --- | --- |
| 함수명 | 구현 내용 |
| decreaseBattery( ) | 함수 인자 설명: 없음  리턴값: 없음  동작 설명  **배터리를 감소시키는 기능을 수행합니다.**  배터리 칸 요소를 가져와서 setInterval을 통해 일정 시간 마다 배터리 칸 내부의 배터리 (자식 요소)를 하나씩 제거합니다.  만약 배터리를 제거했을 때 배터리 칸 내부의 자식 요소가 더 이상 존재하지 않는다면 [lockTimeLayout](#lockTimeLayout)() 함수와 [activeTarget](#activeTarget)() 함수를 수행하고 setInterval을 제거합니다. |
| refillBattery( ) | 함수 인자 설명: 없음  리턴값: 없음  동작 설명  **배터리를 재충전하고 시계를 다시 동작 시킵니다.**  배터리 칸 요소를 가져와서 자식 요소를 생성해  클래스명을 배터리로 설정한 뒤 배터리 칸에 추가합니다.  반복문을 통해 5회 반복합니다.  이후 [unlockTimeLayout](#unlockTimeLayout)() 함수과 [disableTarget](#disableTarget)() 함수 그리고 다시 배터리가 감소하도록 [decreaseBattery](#decreaseBattery)() 함수를 실행합니다. |

파일 명 : timer.js

|  |  |
| --- | --- |
| 함수명 | 구현 내용 |
| setTime() | 함수 인자 설명: 없음  리턴값: 없음  동작 설명  **현재 날짜 및 시간을 설정합니다.**  Date 객체를 통해 현재 날짜 및 시간을 가져와 date, week, hour, minute, second를 설정합니다.  week의 경우 세 글자 영어로 변환합니다.  이후 [setTimeLayout](#setTimeLayout)() 함수를 실행시키면서 date, week, hour, minute, second를 인자로 넘겨줍니다. |
| setTimeLayout(date, week, hour, minute, second) | 함수 인자 설명:  date: 날짜 (연도/월/일)  week: 요일  hour: 시  minute: 분  second: 초  리턴값: 없음  동작 설명  **시간을 표시하는 화면의 레이아웃을 구성합니다.**  각 화면의 날짜와 시간 요소를 가져와 날짜의 경우에는 date와 week로 설정하고  시간의 경우 hour/minute/second를 각각 디지털 숫자로  표현하기 위해 [addDigitNumber](#addDigitNumber)() 함수를 수행합니다.  이때 시간과 시간을 표시하는 요소를 인자로 전달합니다.  setInterval을 통해 매초마다 second를 1씩 증가시키고, second가 60이 되면 minute를 1씩 증가시킨 뒤 0이 됩니다. 마찬가지로 minute도 60이 되면 hour를 1씩 증가시키고 0이 됩니다. hour의 경우 24가 되면 다시 0이 됩니다.  각각 변화가 일어날 때마다 [changeDigitNumber](#changeDigitNumber)() 함수를 통해 디지털 숫자도 변경됩니다. 이때 시간과 시간을 표시하는 요소를 인자로 전달합니다. |
| addDigitNumber(time, digit1, digit2) | 함수 인자 설명:  time: 전달받은 시간 (시/분/초)  digit1: 시간을 디지털 숫자로 표시하는 첫 번째 요소  digit2: 시간을 디지털 숫자로 표시하는 두 번째 요소  리턴값: 없음  동작 설명  **최초의 디지털 숫자 클래스를 추가합니다.**  시간은 일반적으로 두 자리씩 표기되므로 각 시/분/초의 자리를 담당하는 두 개의 요소를 인자로 받습니다.  이후 전달받은 시간을 한 자리 단위로 쪼개어 각 숫자에 맞는 디지털 숫자 class를 각 요소에 추가해줍니다.  ex) time으로 ‘12’를 전달 받으면 ‘1’, ‘2’로 쪼갠 뒤 ‘1’을 디지털 숫자로 표시하는 class와 ‘2’를 디지털 숫자로 표시하는 class를 digit1, digit2 요소에 추가합니다. |
| changeDigitNumber(time, digit1, digit2) | 함수 인자 설명:  time: 전달받은 시간 (시/분/초)  digit1: 시간을 디지털 숫자로 표시하는 첫 번째 요소  digit2: 시간을 디지털 숫자로 표시하는 두 번째 요소  리턴값: 없음  동작 설명  **디지털 숫자 클래스를 업데이트합니다.**  digit1, digit2의 요소가 기존에 갖고 있던 디지털 숫자 class를 제거하고 전달 받은 time에 맞는 디지털 숫자 class를 추가합니다. |

파일 명 : utils.js

|  |  |
| --- | --- |
| 함수명 | 구현 내용 |
| unlockTimeLayout() | 함수 인자 설명: 없음  리턴값: 없음  동작 설명  **시간, 알람 화면을 보이게 합니다.**  시간과 알람을 표시하는 화면 요소에 화면을 안보이게 하는 클래스 “time—off”를 제거합니다. |
| lockTimeLayout() | 함수 인자 설명: 없음  리턴값: 없음  동작 설명  **시간, 알람 화면을 안 보이게 합니다.**  시간과 알람을 표시하는 화면 요소에 화면을 안보이게 하는 클래스 “time—off”를 추가합니다. |
| activeTarget(targetSelector) | 함수 인자 설명:  tagetSelector: 선택된 요소 대상  리턴값: 없음  동작 설명  **버튼을 누를 수 있게 합니다.**  tagetSelector 에 disable 속성과 클래스를 제거합니다. |
| disableTarget(targetSelector) | 함수 인자 설명:  tagetSelector: 선택된 요소 대상  리턴값: 없음  동작 설명  **버튼을 누를 수 없게 합니다.**  tagetSelector 에 disable 속성과 클래스를 추가합니다. |

파일 명 : alarm.js

|  |  |
| --- | --- |
| 함수명 | 구현 내용 |
| changeBtnClass(addBtn,alarmBtns) | 함수 인자 설명:  addBtn: 알람을 추가 및 설정할 수 있는 버튼 요소  alarmBtns: “btn\_\_alarm” 클래스을 지닌 알람과 관련된 모든 버튼 요소들  리턴값: 없음  동작 설명  **알람 설정 모드를 활성화/비활성화합니다.**  addBtn을 클릭 했을 때 “alarm-mode” 클래스를 가지고 있지 않다면 alarmBtns의 모든 요소에 “alram-mode” 클래스를 추가해줍니다. 현재 설정 중인 알람을 표시해주는 요소에 깜빡임 기능을 추가합니다.  알람 생성을 위해 [createAlarm](#createAlarm)()을 수행합니다.  만약 없다면 반대로 제거합니다. 깜빡임도 제거합니다. |
| createAlarm() | 함수 인자 설명: 없음  리턴값: 없음  동작 설명  **알람을 추가합니다.**  알람 요소를 생성해 알람 목록의 가장 위에 추가합니다.  알람 목록의 길이가 3이면 가장 마지막에 위치한 알람 요소(가장 처음에 설정한 알람)를 제거합니다. |
| removeAlarm() | 함수 인자 설명: 없음  리턴값: 없음  동작 설명  **알람을 제거합니다.**  알람 목록에 알람 요소가 존재할 시 가장 위에 있는 알람부터 제거합니다. |
| setAlarm(n) | 함수 인자 설명:  n: 알람 시간을 시/분/초를 담당하는 자리  리턴값: 없음  동작 설명  **알람의 시/분/초를 증가시킵니다.**  알람 시/분/초를 설정하기 위해 각 자리의 요소를 가져와 숫자를 1증가시킵니다.    만약 전달받은 자리가 시(hour)일 시 23을 넘기면 다시 0이 됩니다. 분(minute)과 초(second)일 시 59를 넘기면 다시 0이 됩니다. 시/분/초는 각각 두 자리를 유지합니다. |

파일 명 : buttonEvent.js

|  |  |
| --- | --- |
| 함수명 | 구현 내용 |
| addButtonEvents() | 함수 인자 설명: 없음  리턴값: 없음  동작 설명  **각 버튼에 이벤트 핸들러를 추가합니다.**  배터리 충전, 알람 설정/제거를 각 버튼에 추가합니다. |

파일 명 : main.js

|  |  |
| --- | --- |
| 함수명 | 구현 내용 |
| main() | 함수 인자 설명: 없음  리턴값: 없음  동작 설명  init() 함수를 실행합니다. |
| init() | 함수 인자 설명: 없음  리턴값: 없음  동작 설명  **최초의 화면을 구성하고 기능을 활성화합니다.**  시간 설정, 배터리 감소 시작, 버튼 별 이벤트 추가를 위에 선언된 함수들을 통해 실행합니다.   * [setTime](#setTime)() * [decreaseBattery](#decreaseBattery)() * [addButtonEvents](#addButtonEvents)(); |

# 테스트 결과

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 번호 | 테스트 제목 | 테스트 절차 | 관련 FR | 테스트 결과(O,X) |
| 1 | 시간 표시 | 1. 1초마다 시간이 진행되어 시간 표시 화면이 업데이트 된다.  2. 각 시/분/초에 맞는 디지털 숫자가 표기된다.  3. 초가 59초를 넘기면 분을 증가시키고 0이 된다.  4. 분이 59분을 넘기면 시를 증가시키고 0이 된다.  5. 시가 23을 넘기면 0이 된다. |  | O |
| 2 | 배터리 기능: 기본 | 1. 배터리가 20초마다 한 칸씩 감소한다.  2. 배터리가 0이 되면 시간, 알람 표시 화면이 꺼지며 전원 버튼이 빨간색이 된다.  3. 전원 버튼 클릭 시 배터리가 충전이 되며 시간, 알람 표시 화면이 다시 켜지며 전원 버튼이 초록색이 된다. | FR 1, FR 2,  FR 5 | X (FR 2) |
| 3 | 알람 추가: 최초 등록 | 1. (+) 버튼을 누른다.  2. 알람 목록에 첫 알람 “00:00:00”이 추가된다.  2. 첫 알람이 화면에서 깜빡이는지 확인한다.  3. H/M/S를 눌러 각 자리의 숫자를 설정한다.  4. 숫자를 설정하고 다시 (+) 버튼을 누른다. | FR 3 | O |
| 4 | 알람 추가: 3개 이상 등록 시 | 1. 테스트 3을 반복해 알람을 3개 등록한다.  2. 이 상태에서 알람을 등록할 때 가장 먼저 등록된 알람이 없어지는지 확인한다. | FR 4 | O |
| 5 | 알람 제거 | 1. (-) 버튼을 눌러 가장 위에 위치한 알람이 사라지는 지 확인한다. | FR 6 | O |

# 집중 피드백 받고 싶은 사항

|  |  |
| --- | --- |
| 번호 | 내용 |
| 1 | 순수 javascript만으로는 보통 상태관리를 어떻게 하는지 알고 싶습니다. 저는 특정 요소를 지속적으로 호출하여 체크하거나, 특정 클래스를 추가 및 삭제하는 방식을 사용했는데 이 방식이 효율적인지 잘 모르겠습니다. |
| 2 | 시간을 Digit 숫자를 표시하는데에 어려움을 꽤 겪었습니다. 위에 명시된 내용과 같이 반복적으로 class를 추가하고 제거하는 방식을 통해 화면에 표시했는데 이 방식의 효율 여부와 더 효율적인 방식이 있다면 어떠한 방식이 있는지 궁금합니다. |
| 3 | scss를 처음 사용해 보았습니다. 현업과 비교해보았을 때, 제대로 활용하고 있는지 궁금합니다. |