ATLAS annotation 메뉴창

Todo Plan

1. 메뉴 선택 UI (드롭다운 & 팝업 형식으로 적합한 UI 선택)
2. 전체적으로 상태 관리를 통합해서 init 상태를 추가하고 초기화 버튼 클릭 시 초기 상태로 변경
3. 줌 사이즈나 버튼 등 사라지지 않고 있는 항목들 처리

상태 관리

부모 요소에서 전역으로 상태를 초기화하고 자식 요소들이 해당 상태를 업데이트 할 수 있도록 Context API를 활용 (props drilling 방지)

```
// 부모 컴포넌트 (Viewer)

export interface ViewerValue {
  showScalebar?: boolean;
  useRotationSlider?: boolean;
  useFiltering?: boolean;
  useCapture?: boolean;
  showNavigator?: boolean;
  useICC?: boolean;
  showAnnotation?: boolean;
};

const [viewerValue, setViewerValue] = useState<ViewerValue>({
    showScalebar: true,
    useRotationSlider: true,
    useFiltering: true,
    useCapture: true,
    showNavigator: true,
    useICC: true,
    showAnnotation: true
});

return (
    <ViewerContext.Provider value={{viewerValue, setViewerValue}}>
        ...
    </ViewerContext.Provider>
)
```

```
// 자식컴포넌트 ( Provider로 감싸진 children )
const { viewerValue, setViewerValue } = useContext(ViewerContext);

// ex) viewerValue.showScalebar -> true
```

Toggle 관리를 위한 메뉴창 UI 컴포넌트 추가

antd - `popover` 사용했습니다.

```
import { MenuOutlined } from '@ant-design/icons';
import { Button, Popover, Switch } from 'antd';
import { useContext } from 'react';
```

```
import { ViewerContext } from '../../page/Viewer';
import styled from 'styled-components';

const AnnotationToggleManager = () => {
  /* context state */
  const { viewerValue, setViewerValue } = useContext(ViewerContext);

  /* handler fn */
  const handleShowScalebar = (checked: boolean) => {
    setViewerValue({ ...viewerValue, showScalebar: checked });
  };
  ...
  const handleShowAnnotation = (checked: boolean) => {
    setViewerValue({ ...viewerValue, showAnnotation: checked });
  };

  // 전체 UI 한번에 on/off 할 수 있도록 추가 (선택)
  const handleAllVisible = (checked: boolean) => {
    setViewerValue({
      showScalebar: checked,
      ...
      showAnnotation: checked
    });
  };

  const popoverRender = () => {
    return (
      <PopoverRender>
        <section>
          <span>Show All</span>
          <Switch
              defaultChecked
              checked={
                  Object.values(viewerValue).some(checked => checked === false) ? false : true}
              onChange={handleAllVisible}
             />
        </section>
        ...
        <section>
          <span>Show Annotations</span>
          <Switch
              defaultChecked
                checked={viewerValue?.showAnnotation}
                onChange={handleShowAnnotation}
               />
        </section>
      </PopoverRender>
    )
  }

  return (
    <>
      <Popover content={popoverRender} trigger={['hover']}>
        <Button>
          <MenuOutlined />
        </Button>
      </Popover>
    </>
  )
```

```
}

export default AnnotationToggleManager;


const PopoverRender = styled.div`
  display: flex;
  flex-direction: column;
  gap: 0.5rem;
  min-width: 13rem;

  section {
    display: flex;
    justify-content: space-between;
    align-items: center;
  }

  section:not(:last-child) {
    padding: 0 0 0.5rem 0;
    border-bottom : 1px solid #e8e8e8;
  }

  section:first-child {
    border-bottom : 1px solid black;
  }
`;
```

추가적인 내용

1. Annotation 표시 여부는 단순하게 API 호출 로직에서 처리했습니다.

```
    if (viewerValue?.showAnnotation) {
      const data = await getSlideData(currentSlide.slideId);
      setCurrentSlide(data);
    } else {
      return null;
    }
```

   annotation을 숨기기 하면 openseadragon을 통해 화면에 그릴 annotation 데이터 호출을 막아 화면에 그리지 않도록 조건 처리.

2. PopoverRender(styled-components) & popoverRender(DOM render 함수)

```
만약 popoverRender 추후에 컴포넌트화 하려면 네이밍 수정 필요합니다.
 (지금은 하나의 컴포넌트 파일에서 사용중이라 이렇게 했는데 모듈화 하면서 코드 분리하려면 네이밍 겹쳐서 수정 필요!
```