react实现pdf在线预览

一、react-pdf

`react`实现`pdf`文档可翻页查看，可放大缩小及全屏等功能，选的是`react-pdf`插件

插件引用

1、安装插件

```
bash
复制代码$ npm install react-pdf
# or
$ yarn add react-pdf
```

2、组件引用

```
import { Document, Page, pdfjs } from "react-pdf";
pdfjs.GlobalWorkerOptions.workerSrc =
`//cdnjs.cloudflare.com/ajax/libs/pdf.js/${pdfjs.version}/pdf.worker.js`;
```

3、简单使用

```
<Document
  file="somefile.pdf"
  onLoadSuccess={onDocumentLoadSuccess}
>
  <Page pageNumber={pageNumber} />
</Document>
<p>Page {pageNumber} of {numPages}</p>
```

实现功能

分页功能主要是修改`Page`组件参数来达到想要的效果

```
<Page
pageNumber={pageNumber}
width={pageWidth}
/>
```

1、翻页、跳到指定页面

修改`Page`的`pageNumber`属性值，加一些特定约束，比如已经翻到了第一页或者最后一页，输入的数值最小为1，最大为pdf的总页数值

2、放大缩小、一键适应窗口

修改`Page`的`width`属性值，如果单纯放大缩小，也可以用`scale`属性，不过跟宽度同时用会有点乱，索性只用了宽度

放大缩小过程中发现问题，页面居中用的`flex`布局，`pdf`放大溢出后，无法通过滚动条完全展示

子元素加了样式，已修复

```
width:max-content;
max-width:100%;
```

3、最终源码

样式：

```
css
.view {
  background:#444;
  display: flex;
  justify-content: center;
  height: 100vh;
  padding: 50px 0;
  overflow: auto;
}
.pageContainer {
  box-shadow: rgba(0, 0, 0, 0.2) 0px 2px 4px 0px;
  width:max-content;
  max-width:100%;
}
.pageTool{
  position: absolute;
  bottom: 20px;
  background: rgb(66, 66, 66);
  color: white;
  padding: 8px 15px;
  border-radius: 15px;
  i{
    padding: 5px;
    margin:0 5px;
    &:hover{
      background: #333;
    }
  }
  input{
    display: inline-block;
    width: 50px;
```

```
    text-align: center;
    margin-right: 10px;
    height: 24px;
  }
  input::-webkit-outer-spin-button,
  input::-webkit-inner-spin-button {
    -webkit-appearance: none;
  }
  input[type="number"]{
    -moz-appearance: textfield;
  }
}
```

js:

```
js
import React, { PureComponent } from 'react';
import { Icon, Spin, Tooltip,Input } from 'antd';
import styles from './File.less';

import { Document, Page, pdfjs } from "react-pdf";
pdfjs.GlobalWorkerOptions.workerSrc =
`//cdnjs.cloudflare.com/ajax/libs/pdf.js/${pdfjs.version}/pdf.worker.js`;

class File extends PureComponent {

  state = {
    pageNumber: 1,
    pageNumberInput: 1,
    pageNumberFocus: false,
    numPages: 1,
    pageWidth: 600,
    fullscreen: false
  };

  onDocumentLoadSuccess = ({ numPages }) => {
    this.setState({ numPages: numPages })
  }

  lastPage = () => {
    if (this.state.pageNumber == 1) {
      return
    }
    const page = this.state.pageNumber - 1
    this.setState({ pageNumber: page ,pageNumberInput:page})
  }
  nextPage = () => {
    if (this.state.pageNumber == this.state.numPages) {
      return
```

```
    }
    const page = this.state.pageNumber + 1
    this.setState({ pageNumber: page ,pageNumberInput:page})
  }
  onPageNumberFocus = e => {
    this.setState({ pageNumberFocus: true })
  };
  onPageNumberBlur = e => {
    this.setState({ pageNumberFocus: false
,pageNumberInput:this.state.pageNumber})
  };
  onPageNumberChange = e => {
    let value=e.target.value
    value=value<=0?1:value;
    value=value>=this.state.numPages?this.state.numPages:value;
    this.setState({ pageNumberInput: value })
  };
  toPage = e => {
    this.setState({ pageNumber: Number(e.target.value) })
  };

  pageZoomOut = () => {
    if (this.state.pageWidth <= 600) {
      return
    }
    const pageWidth = this.state.pageWidth * 0.8
    this.setState({ pageWidth: pageWidth })
  }
  pageZoomIn = () => {
    const pageWidth = this.state.pageWidth * 1.2
    this.setState({ pageWidth: pageWidth })
  }

  pageFullscreen = () => {
    if (this.state.fullscreen) {
      this.setState({ fullscreen: false, pageWidth: 600 })
    } else {
      this.setState({ fullscreen: true, pageWidth: window.screen.width -
40 })
    }
  }

  render() {
    const { pageNumber, pageNumberFocus, pageNumberInput,numPages,
pageWidth, fullscreen } = this.state;
    return (
      <div className={styles.view}>
        <div className={styles.pageContainer}>
          <Document
```

```
            file="xxx.pdf"
            onLoadSuccess={this.onDocumentLoadSuccess}
            loading={<Spin size="large" />}
          >
            <Page pageNumber={pageNumber} width={pageWidth} loading={<Spin size="large" />} />
          </Document>
        </div>

        <div className={styles.pageTool}>
          <Tooltip title={pageNumber == 1 ? "已是第一页" : "上一页"}>
            <Icon type="left" onClick={this.lastPage} />
          </Tooltip>
          <Input value={pageNumberFocus ? pageNumberInput : pageNumber}
            onFocus={this.onPageNumberFocus}
            onBlur={this.onPageNumberBlur}
            onChange={this.onPageNumberChange}
            onPressEnter={this.toPage} type="number" /> / {numPages}
          <Tooltip title={pageNumber == numPages ? "已是最后一页" : "下一页"}>
            <Icon type="right" onClick={this.nextPage} />
          </Tooltip>
          <Tooltip title="放大">
            <Icon type="plus-circle" onClick={this.pageZoomIn} />
          </Tooltip>
          <Tooltip title="缩小">
            <Icon type="minus" onClick={this.pageZoomOut} />
          </Tooltip>
          <Tooltip title={fullscreen ? "恢复默认" : '适合窗口'}>
            <Icon type={fullscreen ? "fullscreen-exit" : 'fullscreen'} onClick={this.pageFullscreen} />
          </Tooltip>
        </div>
      </div>
    );
  }
}

export default props => (
  <File {...props} />
);
```

二、iframe

描述：

为了兼容旧版本浏览器，react-pdf在低版本浏览器上兼容性不好，经过多番尝试，选择使用iframe的方式加载显示iframe

显示pdf

第一种：ifreme+src

注意：path必须增加时间戳，否则加载会出问题。

```
<iframe
  key={path}
  src={path + '#scrollbars=0&toolbar=0&statusbar=0'} //文件地址
  title="pdf预览"
  frameBorder="no"
  style={{ width: '100%', height: '100%' }}
/>
```

- pdf预览使用的iframe，由于X-Frame-Options的问题，无法直接预览，所以使用代理解决，即：部署的时候需要在nginx增加如下配置，实际地址为接口返回的host部分，代码中会将此部分替换为/fileApi，所以会命中规则。

```

location ^~/services/ {

      proxy_hide_header X-Frame-Options;

      add_header X-Frame-Options SAMEORIGIN;

      ...

}
```

第二种：iframe + base64格式编码，以及`URL.createObjectURL()`方法生成的地址。

```
fetch('http://xxx.pdf')
  .then( res => res.blob()) // 解析res值为blob
  .then( response => {
    const url = URL.createObjectURL(response);
    this.setState({
      htmlStr: url
    })
})
```