

jsPdfCmp

준비물

LWC	jsPdfCmp	소스 다운로드 : https://github.com/flashteker/jspdf (static js library는 zip으로 변경 필요)
	commonShowToast	
Apex class	JsPdfController	
	JsPdfController_Test	
Static Resource	pdfviewer.zip	
	jspdf.zip	



Files



main



Go to file



- > .idea
- > config
- ▼ force-app/main/default
 - > aura
 - > classes
 - > lwc
 - > staticresources
- .eslinignore
- .forceignore
- .gitignore
- OrgA.iml
- README.md
- jest.config.js
- jspdf 설명서.pdf
- package.json
- sfdx-project.json

Static Resources

[Help for this Page](#) ?

Use static resources to upload content that you want to reference in a Visualforce page, including .zip and .jar files, images, stylesheets, JavaScript, and other files.

View: All [Create New View](#)

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z Other All

New									
Action	Name	Namespace Prefix	Description	MIME Type	Size	Created By Alias	Created Date	Last Modified Date +	Cache Control
Edit Del	jspdf			application/zip	151,822 file		9/3/2025	9/17/2025, 2:47 PM	Public
Edit Del	pdfviewer			application/zip	4,551,488 file		9/4/2025	9/17/2025, 2:47 PM	Public

view의 type

TYPE	성격	설명
stack	container	content를 위에서 아래로 쌓는다.
horizontal	container	content를 왼쪽에서 오른쪽으로 쌓는다.
table	container	텍스트를 테이블 형식으로 담는다.
text	content	텍스트
image	content	이미지
line	content	라인
page	function	새로운 페이지를 추가한다.

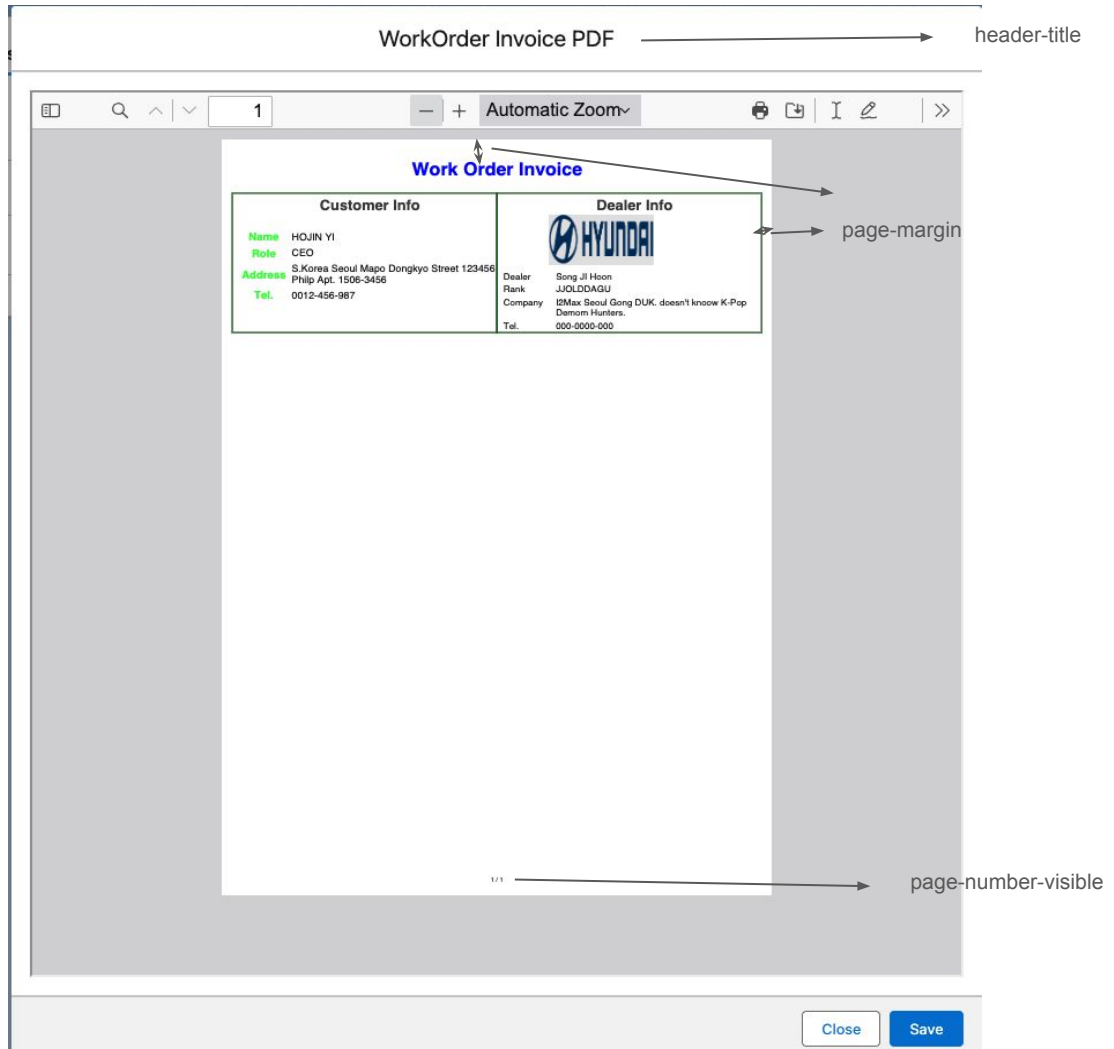
jsPdfCmp 사용 개요 - HTML

HTML

```
<template>
  <c-js-pdf-cmp
    header-title="WorkOrder Invoice PDF"
    ondrawready={handleOnDrawReady}
    record-id={recordId}
    file-name={pdfFileName}
    onclose={handleClose}
    onsave={handleSave}
    page-margin={pdfPageMargin}
    page-number-visible={pageNumberVisible}
  ></c-js-pdf-cmp>
</template>
```

- * header-title : pdf 창의 타이틀을 나타낸다.
- * page-margin : pdf전반의 여백을 지정한다.
- * page-number-visible : 페이지 마다 하단에 페이지 번호를 나타낼지 여부를 지정한다.
- * file-name : 저장시 파일 이름
- * record-id : 저장할 오브젝트의 id
- * onclose : 닫기 버튼이 눌러졌을 때 호출된다.
- * onsave : 저장 버튼이 눌러졌을 때 호출된다.
- * ondrawready : 페이지에 그릴 준비가 완료되면 호출 된다.

file-name과 record-id가 있을 경우 "저장" 버튼을 누르면 자동으로 저장한다.



jsPdfCmp 사용 개요 - js controller

js

```
import { LightningElement, api, track, wire } from 'lwc';
import { CloseActionScreenEvent } from 'lightning/actions';
export default class WoPdfCmp extends LightningElement {
  handleClose() {
    this.dispatchEvent(new CloseActionScreenEvent());
  }
  handleSave(event) {
    const kBase64PDF = event.detail.base64PDF;
    //do something for saving
    this.handleClose();
  }
  async handleOnDrawReady(event){
    const kPdfCmp = this.template.querySelector('c-js-pdf-cmp');
    if(!kPdfCmp) return;
    const kBodies = [
      {type:"text", text:"Hello world"}
    ];
    kPdfCmp.startDraw({
      bodies:kBodies,
      footer:null,
      header:null
    })
  }
}
```

```
jsPdfCmp.startDraw({
  bodies:Array,
  footer:Object,
  header:Object
})
```

* 그리기 호출은 jsPdfCmd.startDraw(data)에서 시작한다.

* data에는 bodies/footer/header 3개의 필드가 있다.(footer/header는 이후 설명)

* bodies는 배열 형식이며, view들이 들어갈 수 있다.

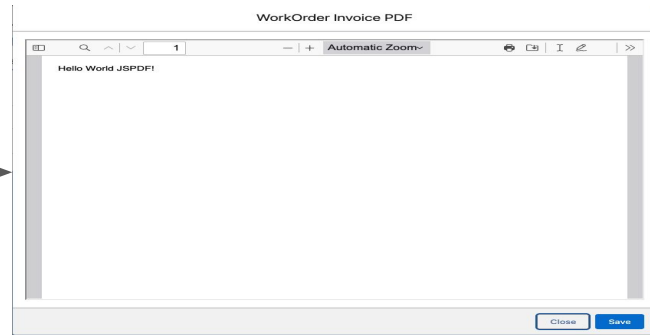
* 샘플 코드에서 알 수 있듯이 view는 특별한 객체를 사용하지 않고, 범용적인 오브젝트에 type으로 view를 결정한다.(그래서 모든 view의 “type”필드 값은 필수)

Sample - text

```
async handleOnDrawReady(event){
  const kPdfCmp = this.template.querySelector('c-js-pdf-cmp');
  const kBodies = [
    {type:"text", text : "Hello World JSPDF!"}
  ];
  kPdfCmp.startDraw({
    bodies:kBodies
  })
}
```

* 텍스트에는 텍스트에 관한 다양한 속성들이 가능하다.

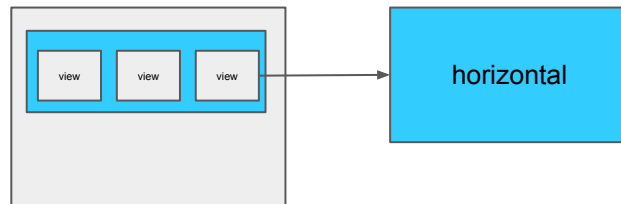
```
async handleOnDrawReady(event){
  const kPdfCmp = this.template.querySelector('c-js-pdf-cmp');
  const kBodies = [
    {type:'text',
      text:'Hello World JSPDF!!',
      styles:{
        halign:"center",
        fontSize:20,
        fontStyle:"bold",
        color:{r:0,g:0,b:250}
      },
      margin:{top:4}
    }
  ];
  kPdfCmp.startDraw({
    bodies:kBodies
  })
}
```



Sample (Layout) - horizontal

Horizontal은 좌에서 우로 뷰를 나열하는 레이아웃이다.

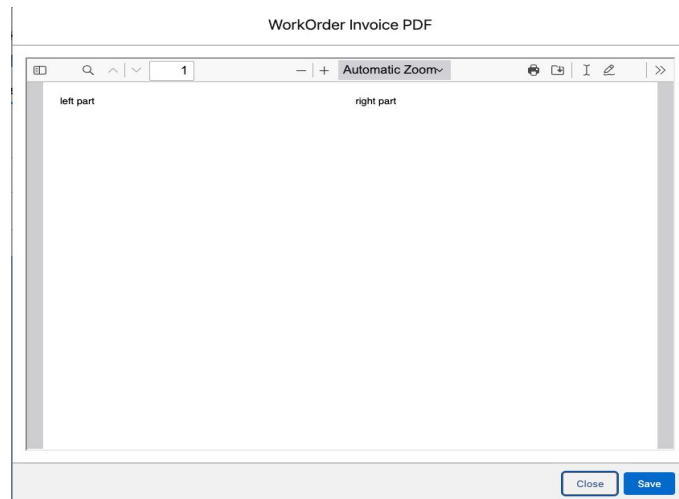
Layout(horizontal/stack)은 **children**이라는 배열을 가지며, 이 배열에는 뷰들이 나열된다.



```
async handleOnDrawReady(event){
  const kPdfCmp = this.template.querySelector('c-js-pdf-cmp');
  const kBodies = [
    {
      type:'horizontal',

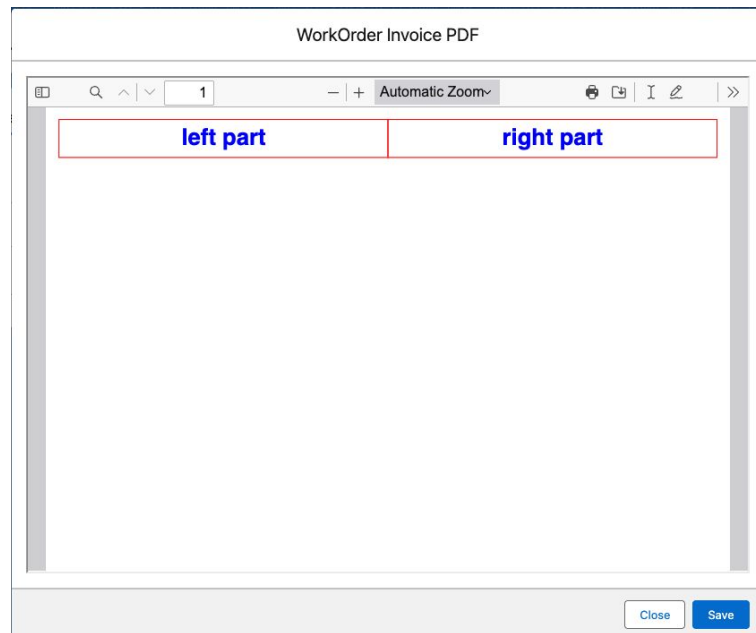
      children:[
        { type:"text", text:"left part"},
        {type:"text",text:"right part" }
      ]
    }
  ];
  kPdfCmp.startDraw({
    bodies:kBodies
  })
}
```

* horizontal layout에도 여러가지 속성을 지정할 수 있다.



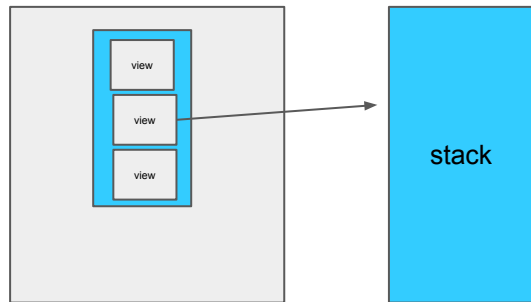
Sample - horizontal

```
async handleOnDrawReady(event){
  const kPdfCmp = this.template.querySelector("c-js-pdf-cmp");
  const kBodies = [
    {
      type:'horizontal',
      border:{thick:0.2,color:{r:255}},
      children:[
        {
          type:"text",
          text:"left part",
          styles:{
            halign:"center",
            fontSize:20,
            fontStyle:"bold",
            color:{r:0,g:0,b:250}
          }
        },
        {
          type:"text",
          text:"right part",
          styles:{
            halign:"center",
            fontSize:20,
            fontStyle:"bold",
            color:{r:0,g:0,b:250}
          }
        }
      ]
    }
  ];
  kPdfCmp.startDraw({
    bodies:kBodies
  })
}
```

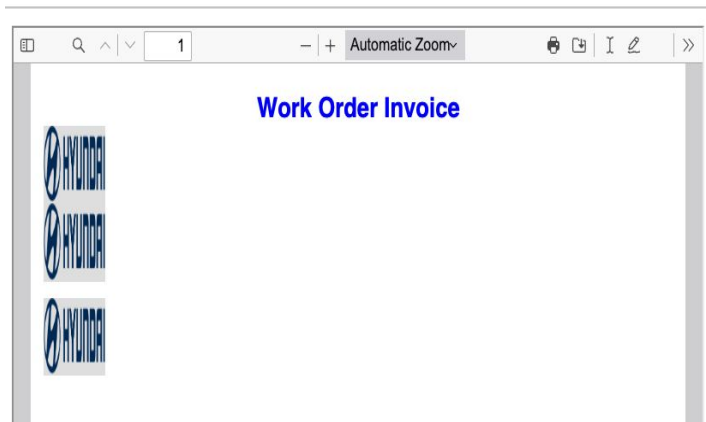


Sample(Stack) - stack

* **stack**은 위에서 아래로 차곡차곡 뷰들을 나열한다.



```
async handleOnDrawReady(event){
  this.logoBase64 = await this.preloadImage(hyundai_logo);
  const kPdfCmp = this.template.querySelector('c-js-pdf-cmp');
  const kBodies = [
    {type:'stack',
     children:[
       this.getHeaderTitle(),
       this.getImageData(20,20),
       this.getImageData(20,20),
       this.getImageData(20,20, {top:2})
     ]
    }
  ];
  kPdfCmp.startDraw({bodies:kBodies})
}
```



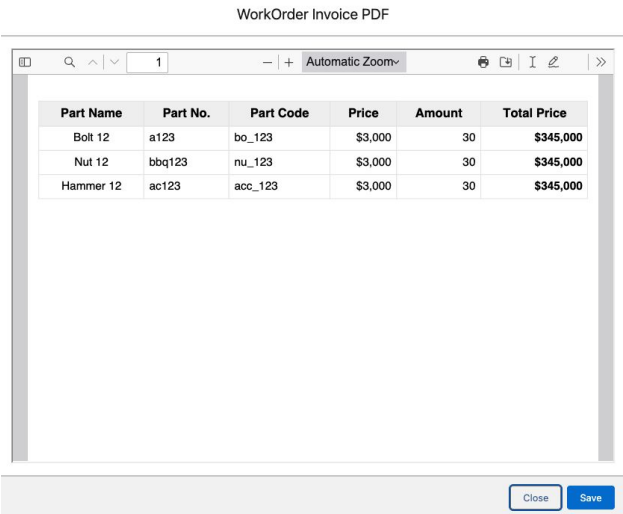
(!Tip) View data를 생성할 때 메소드를 이용하기를 권장한다. 이럴 경우 재사용이나 코드 관리가 편리하고, 뷰 구조 파악에 용이하다.

Sample - table

- table은 순수하게 텍스트를 그리드 형식으로 표시하기 위한 뷰이다.(이미지나 다른 뷰 불가)
- 각 셀마다 성질을 달리 줄 수 있게 하기 위하여 데이터들이 비교적 복잡하지만, head나 body를 제외하고는 모두 기본값으로 처리할 수 있다.

```
async handleOnDrawReady(event){
  const kPdfCmp =
this.template.querySelector('c-js-pdf-cmp');
  const kBodies = [
    this.getTableView()
  ];
  kPdfCmp.startDraw({
    bodies:kBodies
  })
}

getTableView(){
  const kHeadBgColor = {r:240, g:240, b:240}
  const kHeadStyles = [
    {halign:"center", fontSize:13, bgColor:kHeadBgColor},
    {halign:"center", fontSize:13, bgColor:kHeadBgColor},
    {halign:"center", fontSize:13, bgColor:kHeadBgColor},
    {halign:"center", fontSize:13, bgColor:kHeadBgColor},
    {halign:"center", fontSize:13, bgColor:kHeadBgColor},
    {halign:"center", fontSize:13, bgColor:kHeadBgColor},
  ]
  const kBodyStyles = [
    {halign:"center", fontSize:12},
    {halign:"left", fontSize:12},
    {halign:"left", fontSize:12},
    {halign:"right", fontSize:12},
    {halign:"right", fontSize:12},
    {halign:"right", fontSize:12, fontStyle:"bold"},
  ];
  return{
    type:'table',
    border:{thick:0.2, color:{r:230, g:230, b:230}},
    margin:{top:6},//space from parent
    cellPadding:2,//셀과 텍스트간의 간격 head, body 구분하지 않음
    headStyles:kHeadStyles,
    bodyStyles:kBodyStyles,
    head:['Part Name','Part No.', 'Part Code', 'Price', 'Amount', 'Total Price'],
    body:[
      ['Bolt 12', 'a123', 'bo_123', '$3,000', '30','$345,000'],
      ['Nut 12', 'bbq123', 'nu_123', '$3,000', '30','$345,000'],
      ['Hammer 12', 'ac123', 'acc_123', '$3,000', '30','$345,000']
    ]
  }
}
```



Sample - 헤더가 없는 table

- head를 지정하지 않으면 헤더 없이 그려진다.(이 원리를 이해하면 다양한 형식의 표를 만들 수 있다.)

```
async handleOnDrawReady(event){
  const kPdfCmp = this.template.querySelector('c-js-pdf-cmp');
  const kBodies = [
    this.getTableView(),
    this.getSummaryTableData()
  ];
  kPdfCmp.startDraw({
    bodies:kBodies
  })
}

.....
getSummaryTableData(){
  const kBodyStyles = [
    {fontSize:12, color:{b:255}},
    {halign:"right", fontSize:12, color:{r:255}}
  ];
  return{
    type:'table',
    margin:{left:100},
    border:{thick:0.2, color:{r:200, g:200, b:0}},
    bodyStyles:kBodyStyles,
    body:[
      ["Total Price", "$1,000,000"],
      ["Total TAX", "$4,000"],
      ["Total Amount", "$1,040,000"]
    ]
  }
}
```



Part Name	Part No.	Part Code	Price	Amount	Total Price
Bolt 12	a123	bo_123	\$3,000	30	\$345,000
Nut 12	bbq123	nu_123	\$3,000	30	\$345,000
Hammer 12	ac123	acc_123	\$3,000	30	\$345,000
Total Price					\$1,000,000
Total TAX					\$4,000
Total Amount					\$1,040,000

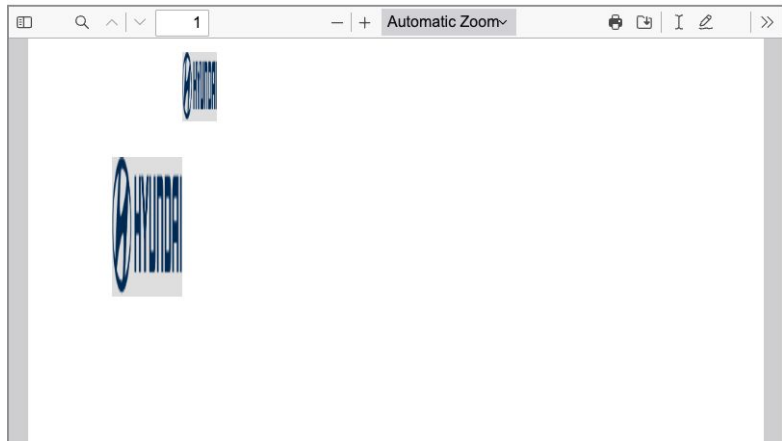
Sample - image

```
async handleOnDrawReady(event){
  this.logoBase64 = await this.preloadImage(hyundai_logo);
  const kPdfCmp = this.template.querySelector('c-js-pdf-cmp');
  const kBodies = [
    this.getImageData(10,20,{left:20, top:0}),
    this.getImageData(20,40,{left:10, top:5})
  ];
  kPdfCmp.startDraw({
    bodies:kBodies
  })
}

async preloadImage(imgUrl) {
  try {
    const response = await fetch(imgUrl);
    const blob = await response.blob();
    const base64 = await new Promise((resolve, reject) => {
      const reader = new FileReader();
      reader.onloadend = () => resolve(reader.result);
      reader.onerror = reject;
      reader.readAsDataURL(blob);
    });
    return base64 ;
  } catch (e) {
    console.error('Failed to load damage legend image:', imgUrl, e);
    return null;
  }
}

getImageData(width, height, margin){
  return {
    type:"image",
    margin:margin,
    image:{src:this.logoBase64, w:width, h:height}
  }
}
```

* image는 base64로 인코딩된 데이터만 사용할 수 있다.



Sample - line

```
async handleOnDrawReady(event){
  this.logoBase64 = await this.preloadImage(hyundai_logo);
  const kPdfCmp = this.template.querySelector('c-js-pdf-cmp');
  const kBodies = [
    this.getImageData(20,20,{left:10, top:0}),
    {
      type:'line',
      border:{thick:0.5, color:{r:0, g:0, b:255}},
      margin:{left:20, top:2},
      width:20
    },
  ];
  kPdfCmp.startDraw({bodies:kBodies})
}
```

* image에서 2 픽셀 아래에 파란색 라인을 그린다

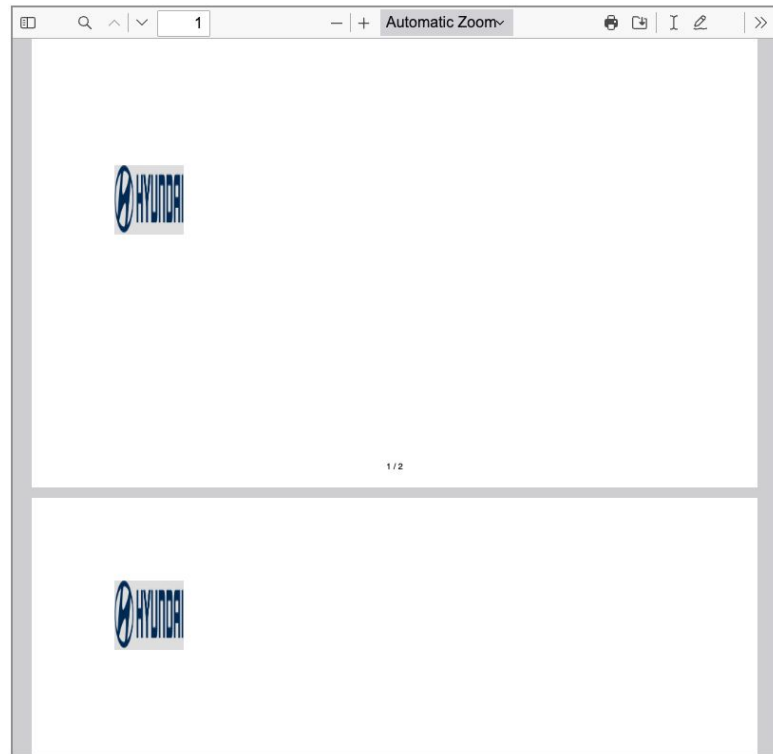


Sample - page

```
async handleOnDrawReady(event){
  this.logoBase64 = await this.preloadImage(hyundai_logo);
  const kPdfCmp = this.template.querySelector('c-js-pdf-cmp');
  const kBodies = [
    this.getImageData(20,20,{left:10, top:100}),
    {type:'page'},
    this.getImageData(20,20,{left:10, top:10}),
  ];
  kPdfCmp.startDraw({bodies:kBodies})
}
```

* 새로운 페이지를 추가한다.

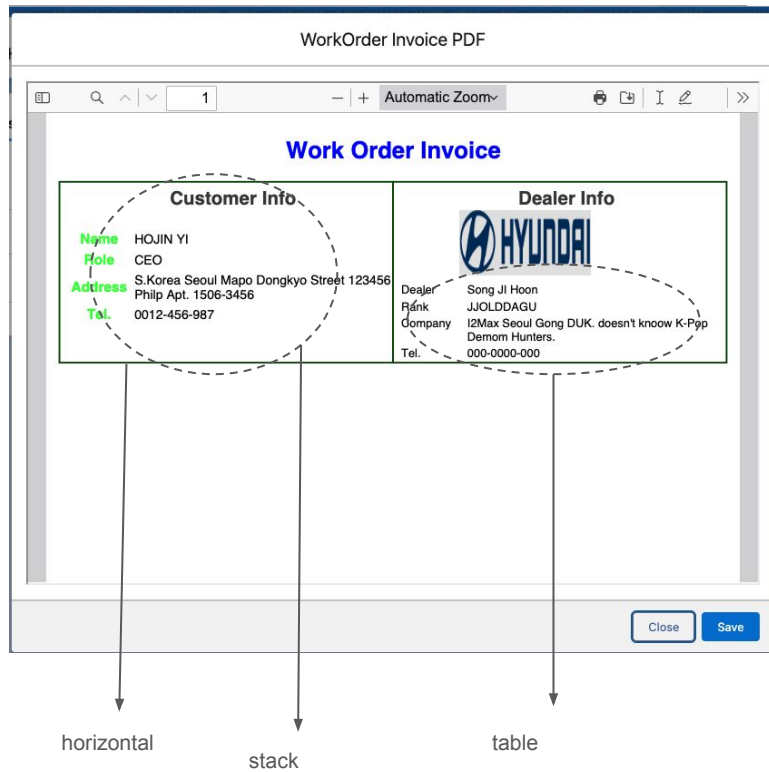
* 최상위 레이아웃에서만 사용할 수 있으며, 내부 레이아웃에서는 사용할 수 없다.(사용하더라도 무시된다.)



Container/Component 모델

* Layout은 또 다른 Layout을 자식뷰로 가질 수 있으며, 이를 이용하여 복잡한 뷰를 생성할 수 있다.

```
async handleOnDrawReady(event){
  this.logoBase64 = await this.preloadImage(hyundai_logo);
  const kPdfCmp = this.template.querySelector('c-js-pdf-cmp');
  const kBodies = [
    this.getHeaderTitle(),
    this.getHeaderView()
  ];
  kPdfCmp.startDraw({bodies:kBodies})
}
...
getHeaderView(){
  return {
    type: 'horizontal',
    margin: {top:5},
    border: {thick:0.5, color:{r:0, g:70, b:0}},
    children:[
      {
        type: 'stack',
        children:[
          this.getTextData('Customer Info', {left:2, top:2},
            {halign:"center",fontSize:16,fontStyle:"bold",color:{r:60,g:60,b:60}}),
          this.getCustomerInfoTableView()
        ]
      },
      {
        type: 'stack',
        children:[
          this.getTextData('Dealer Info', {left:2, top:2},
            {halign:"center",fontSize:16,fontStyle:"bold",color:{r:60,g:60,b:60}}),
          this.getImageData(40,20,{left:10}),
          this.getDealerInfoTableView()
        ]
      }
    ]
  }
}
```



footer

* footer는 제일 마지막 페이지 하단에 표시되는 영역을 나타내는 뷰이다.

* height 및 child 필드를 가지며, child는 view이다.

* 페이지 마다 반복되지 않는다.

```
async handleOnDrawReady(event){
    this.logoBase64 = await this.preloadImage(hyundai_logo);
    const kPdfCmp = this.template.querySelector('c-js-pdf-cmp');
    const kBodies = [
        this.getHeaderTitle(),
        this.getHeaderView()
    ];
    kPdfCmp.startDraw({
        bodies:kBodies,
        footer:{height:45, child:this.getFooterView()}
    })
}
...
getFooterView(){
    return {
        type:'horizontal',
        children:[
            this.getSignatureView('Director'),
            this.getSignatureView('Dealer'),
            this.getSignatureView('Customer')
        ]
    }
}
getSignatureView(title){
    return {
        type:'stack',
        margin:{left:0},
        children : [
            {type:'text', styles:{fontSize:10}, text:"Signed by "},
            this.getImageData(30,25),
            {type:'line', border:{thick:0.5}, margin:{top:0},width:50},
            {type:'text', styles:{fontSize:10}, text:title}
        ]
    }
}
```

Work Order Invoice

Customer Info		Dealer Info	
Name	HOJIN YI		
Role	CEO		
Address	S.Korea Seoul Mapo Dongkyo Street 123456		
Tel.	0012-456-987		
		Dealer	Song Ji Hoon
		Rank	JJOLDDAGU
		Company	BMMax Seoul Gong DUK. doesn't know K-Pop
			Demon Hunters.
		Tel.	000-0000-000

Signed by

Director

Signed by

Dealer

Signed by

Customer

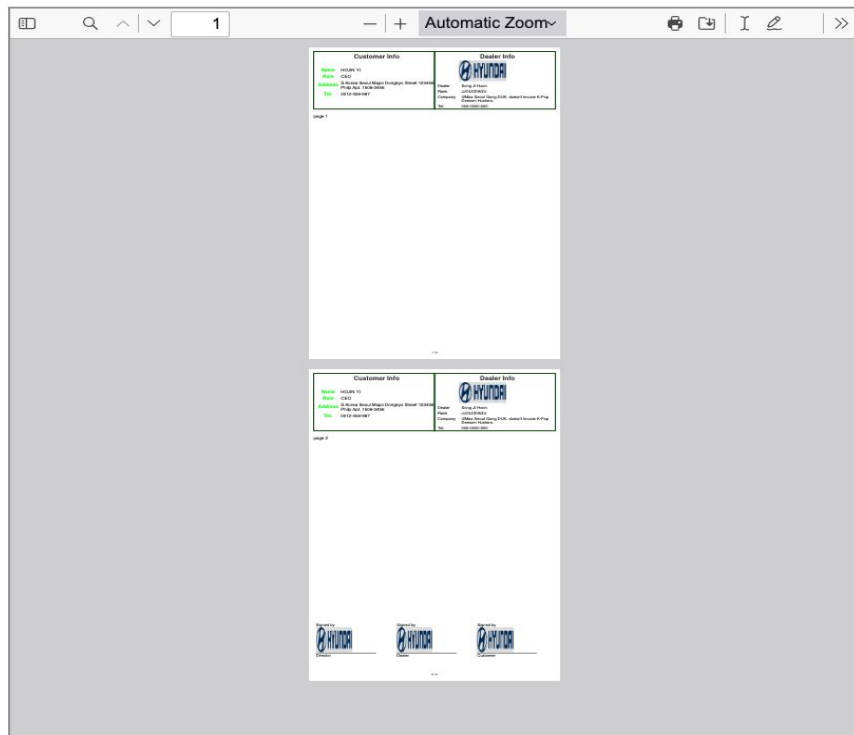
header

* header는 페이지마다 맨 상단에 표시되는 뷰이다.

* footer와 마찬가지로 height와 child속성을 가진다.

* 왜 옵션없이 페이지 마다 반복 되냐규? 반복되지 않는 헤더라면 그냥 **bodies**의 맨 처음 뷰 위치에 놓으면 된다.

```
async handleOnDrawReady(event){
  this.logoBase64 = await this.preloadImage(hyundai_logo);
  const kPdfCmp = this.template.querySelector('c-js-pdf-cmp');
  const kBodies = [
    this.getTextView("page 1"),
    {type:"page"},
    this.getTextView("page 2")
  ];
  kPdfCmp.startDraw({
    bodies:kBodies,
    footer:{height:45, child:this.getFooterView()},
    header:{height:60, child:this.getHeaderView()}
  })
}
```



TIP - 매우 긴 장문을 표시할 때

* jsPdf의 가장 큰 단점 2개

1. 화면에 그린 이후 수정 불가

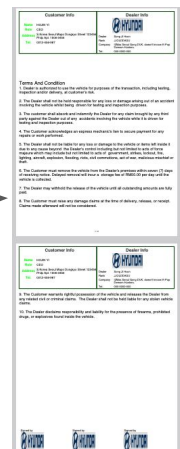
2. header나 footer 영역을 자동으로 제외하고 그리기 불가능

이러다 보니, header가 있을 경우 매우 긴 장문(예를 들어 라이선스 문구)을 기입할 때 많은 여백을 남기고 다음 페이지로 넘어간

그래서 긴문장은 배열로 잘라서 text view로 처리하기를 권장한다.

```
async handleOnDrawReady(event){
  this.logoBase64 = await this.preloadImage(hyundai_logo);
  const kPdfCmp = this.template.querySelector('c-js-pdf-cmp');
  const kBodies = [
    this.getTextView("Terms And Condition",{top:30}, {fontSize:18}),
    this.getTextView(this.txtTermsList.join("\n\n"),{top:20},{fontSize:14})
  ];
  kPdfCmp.startDraw({
    bodies:kBodies,
    footer:{height:45, child:this.getFooterView()},
    header:{height:60, child:this.getHeaderView()}
  })
}
```

```
async handleOnDrawReady(event){
  this.logoBase64 = await this.preloadImage(hyundai_logo);
  const kPdfCmp = this.template.querySelector('c-js-pdf-cmp');
  const kBodies = [
    this.getTextView("Terms And Condition",{top:30}, {fontSize:18})
  ];
  this.txtTermsList.forEach(txt => {
    const kTxtData = this.getTextView(txt+'\n',{top:0},{fontSize:14});
    kBodies.push(kTxtData);
  })
  kPdfCmp.startDraw({
    bodies:kBodies,
    footer:{height:45, child:this.getFooterView()},
    header:{height:60, child:this.getHeaderView()}
  })
}
```



TIP - table view의 경우 레코드(행)가 많아질때

* table은 기본적으로 자동으로 페이지를 나누어준다 하지만 header나 footer가 있을 경우 레코드가 길어 페이지가 나누어질 때 header나 footer(페이지 표시 영역포함) 영역을 침범한다.

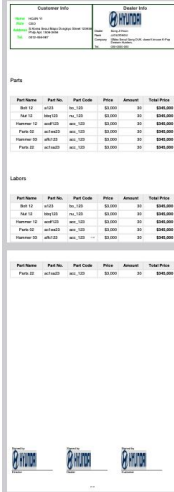
이럴 경우, 명시적으로 페이지를 나누어 (type:"page") 이를 피한다.

```
async handleOnDrawReady(event){
    this.logoBase64 = await this.preloadImage(hyundai_logo);
    const kPdfCmp = this.template.querySelector("c-js-pdf-cmp");
    const kBodies = [
        this.getTextView("Parts",{top:30}, {fontSize:18}),
        this.getTableView(),
        this.getTextView("Labors",{top:30}, {fontSize:18}),
        this.getTableView()
    ];

    kPdfCmp.startDraw({
        bodies:kBodies,
        footer:{height:45, child:this.getFooterView()},
        header:{height:60, child:this.getHeaderView()}
    })
}
```

```
async handleOnDrawReady(event){
    this.logoBase64 = await this.preloadImage(hyundai_logo);
    const kPdfCmp = this.template.querySelector("c-js-pdf-cmp");
    const kBodies = [
        this.getTextView("Parts",{top:30}, {fontSize:18}),
        this.getTableView(),
        {type:"page"},
        this.getTextView("Labors",{top:30}, {fontSize:18}),
        this.getTableView()
    ];

    kPdfCmp.startDraw({
        bodies:kBodies,
        footer:{height:45, child:this.getFooterView()},
        header:{height:60, child:this.getHeaderView()}
    })
}
```



Part Name	Part No.	Part Code	Price	Amount	Total Price
Part 12	12345	12345	\$2,000.00	1	\$2,000.00
Part 13	12345	12345	\$2,000.00	1	\$2,000.00
Part 14	12345	12345	\$2,000.00	1	\$2,000.00
Part 15	12345	12345	\$2,000.00	1	\$2,000.00
Part 16	12345	12345	\$2,000.00	1	\$2,000.00



Part Name	Part No.	Part Code	Price	Amount	Total Price
Part 12	12345	12345	\$2,000.00	1	\$2,000.00
Part 13	12345	12345	\$2,000.00	1	\$2,000.00
Part 14	12345	12345	\$2,000.00	1	\$2,000.00
Part 15	12345	12345	\$2,000.00	1	\$2,000.00
Part 16	12345	12345	\$2,000.00	1	\$2,000.00

범용 속성

Property	Data Type	설명
styles	Object	텍스트 및 font 속성 (halign, valign, fontSize, fontStyle, color)
bodyStyles / headStyles	Object	table view 의 cell의 텍스트 및 font 속성 (halign, valign, fontSize, fontStyle, color, cellWidth)
margin	Object	{left:, top:, right:, bottom:}. 상위 컨테이너 내에서의 자신 위치
border	Object	{thick:, color:}
color	Object	{r, g, b}
fontSize	Number	font size
text	String	“text” view에서 text 내용
left/ top/right/ bottom	Number	상위 뷰에서의 나의 상대적인 위치
halign	String	“left”(default) / “center” / “right”
valign	String	“top” / “middle”(default)/“bottom”
fontStyle	String	“bold”, “normal”(default)
r/g/b	Number	0 ~ 255
width/height	Number	너비 및 높이
cellWidth	Number	table에서 한 컬럼의 너비를 지정한다 .
cellPadding	Number	table cell내의 패딩
body	array	1차 배열 [str, str] . 테이블 헤더의 필드 값들을 나열 . table에서만 사용
head	array	2차 배열 [[str, str...], [str, str...],....]. 각 행들의 집합.. table에서만 사용
image	Object	src, width, height
src	String	image base64 string data

각 View의 속성들 - stack

Property	Data Type	설명
type	String 상수	'stack'
margin	Object	
border	Object	
children	Array	child Views

```
{
  type:'stack',
  margin:{left:2, top:10, right:2, bottom:2},
  border:{thick:0, color:{r:255}},
  children : [
    {type:'text', styles:{fontSize:10}, text:"Signed by "},
    this.getImageData(30,25),
    {type:'line', border:{thick:0.5}, margin:{top:0},width:50},
    {type:'text', styles:{fontSize:10}, text:title}
  ]
}
```

각 View의 속성들 - horizontal

Property	Data Type	설명
type	String 상수	'horizontal'
margin	Object	
border	Object	
children	Array	child Views

```
{
  type:'horizontal',
  margin:{left:2, top:3},
  border:{thick:0.2,color:{r:255}},
  children:[
    this.getSignatureView('Director'),
    this.getSignatureView('Dealer'),
    this.getSignatureView('Customer')
  ]
}
```

각 View의 속성들 - page

Property	Data Type	설명
type	String 상수	'page'

```
{
  type:'page',
}
```


각 View의 속성들 - line

Property	Data Type	설명
type	String 상수	'line'
margin	Object	
border	Object	
width	Number	
height	Number	

```
{
  type:'line',
  border:{thick:0.5},
  margin:{top:0},
  width:50
}
```

각 View의 속성들 - image

Property	Data Type	설명
type	String 상수	'image'
margin	Object	
border	Object	
image	Object	

```
{
  type:"image",
  margin:margin,
  image:{src:this.logoBase64, width:kWidth, height:kHeight}
}
```

각 View의 속성들 - text

Property	Data Type	설명
type	String 상수	'text'
margin	Object	
border	Object	
styles	Object	
text	String	

```
{
  type:'text',
  styles:{fontSize:10},
  text:"Signed by "
}
```

각 View의 속성들 - table

Property	Data Type	설명
type	String 상수	'table'
margin	Object	
border	Object	
bodyStyles	Object	
headStyles	Object	
cellPadding	Number	
head	Array	1차 배열 [str, str] . 테이블 헤더의 필드 값들을 나열
body	Array	2차 배열 [[str, str...], [str, str...],....]. 각 행들의 집합.

```
const kHeadBgColor = {r:240, g:240, b:240}
const kHeadStyles = [
  {halign:"center", fontSize:14, bgColor:kHeadBgColor},
  {halign:"center", fontSize:14, bgColor:kHeadBgColor},
  {halign:"center", fontSize:14, bgColor:kHeadBgColor},
  {halign:"center", fontSize:14, bgColor:kHeadBgColor},
  {halign:"center", fontSize:14, bgColor:kHeadBgColor},
  {halign:"center", fontSize:14, bgColor:kHeadBgColor},
]
const kBodyStyles = [
  {halign:"center", fontSize:14},
  {halign:"left", fontSize:14},
  {halign:"left", fontSize:14},
  {halign:"right", fontSize:14},
  {halign:"right", fontSize:14},
  {halign:"right", fontSize:14, fontStyle:"bold"},
];
```

```
{
  type:'table',
  border:{thick:0.2, color:{r:230, g:230, b:230}},
  margin:{top:16},//space from parent
  cellPadding:2,//셀과 텍스트간의 간격 head, body 구분하지 않음
  headStyles:kHeadStyles,
  bodyStyles:kBodyStyles,
  head:['Part Name','Part No.', 'Part Code', 'Price', 'Amount', 'Total Price'],
  body:[
    ['Bolt 12', 'a123', 'bo_123', '$3,000', '30','$345,000'],
    ['Nut 12', 'bbq123', 'nu_123', '$3,000', '30','$345,000']
  ]
}
```

기타 - 창 닫기 및 저장하기

jsPdfCmp에 등록된 이벤트 핸들러를 이용한다.

```
<template>
  <c-js-pdf-cmp
    header-title="WorkOrder Invoice PDF"
    ondrawready={handleOnDrawReady}
    record-id={recordId}
    file-name={pdfFileName}
    onclose={handleClose}
    onsave={handleSave}
    page-margin={pdfPageMargin}
    page-number-visible={pageNumberVisible}
  ></c-js-pdf-cmp>
</template>
```

```
import { CloseActionScreenEvent } from 'lightning/actions';

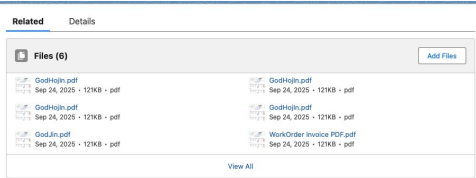
....
handleClose() {
  this.dispatchEvent(new CloseActionScreenEvent());
}

handleSave(event) {
  const kBase64PDF = event.detail.base64PDF;
  //do something for saving
  this.handleClose();
}
```

* 창 닫기는 CloseActionScreenEvent를 이용한다.

* 저장은 event.detail.base64PDF를 이용한다. 이 변수에는 pdf를 base64 string 형식으로 담고 있다.

* 만약 “record-id”와 “file-name”이 지정되면 자동으로 파일을 저장한다.



<https://github.com/flashteker/jspdf>에 접속하면 jsPdfCmp를 이용한 woPdfCmp 소스가 있다. 다양한 View뿐만 아니라, 창 닫기 및 저장하기에 관한 샘플 소스들이 있다.

