

LSWUpDown 생성 및 API 활용 가이드

1. 업로드, 다운로드 창 생성

파라미터는 총 7개이며 0번과 1번 파라미터인 divName과 id는 필수적으로 입력하여야 합니다.

```
LSWUpDownPrototype.prototype.option = { // 객체의 옵션(정보) (유저가 설정 or 기본값)
  divId:"", // 붙힐 div의 id값(필수)
  id:"LSWUpDownDefaultID", // id값(필수)
  width:400, // 창 너비
  height:300, // 창 높이
  option:"both", // 생성하는 옵션 (업로드만, 다운로드만, 둘다)
  bp:0, // 0 = 버튼있음, 1 = 버튼없음(버튼은 알아서 커스텀)
  loaded:function(Iframe){ // 온로드 기본함수, 생성한 Iframe를 제공
    console.log("Default onload message id="+this.id+", option="+this.option);
    console.log(Iframe);
  },

  isIE:false, // IE인지 브라우저 체크
  wl:null, // 파일 업로드 대기목록 (각 id별로 독립된 리스트를 부여합니다.)
  result:null // event 함수용 return을 담는 목록 (각 id별로 독립된 리스트를 부여합니다.)
}
```

프로토 타입의 옵션 구조 및 기본값

```
// **parm{ divId, id, width, height, option, BP, loaded }**
// divId = 생성된 창을 붙힐 div의 id
// id = 생성할 창의 고유 ID (다 독립되어 개별로 작동하게 됩니다.)
// < > 이때 업로드와 다운로드의 아이디가 같으면 새로그침을 동기화하여 편리하게 작동하도록 구현함
// width = 업로더, 다운로드의 가로길이
// height = 업로더, 다운로드의 세로길이
// option : Upload, Download, both(둘다생성, id도같이)
// BP : 0 = 기본값, 1 = 창만(버튼은 API 활용하여 알아서)
// loaded : Iframe이 로드되면 실행될 함수
```

각 파라미터에 대한 설명

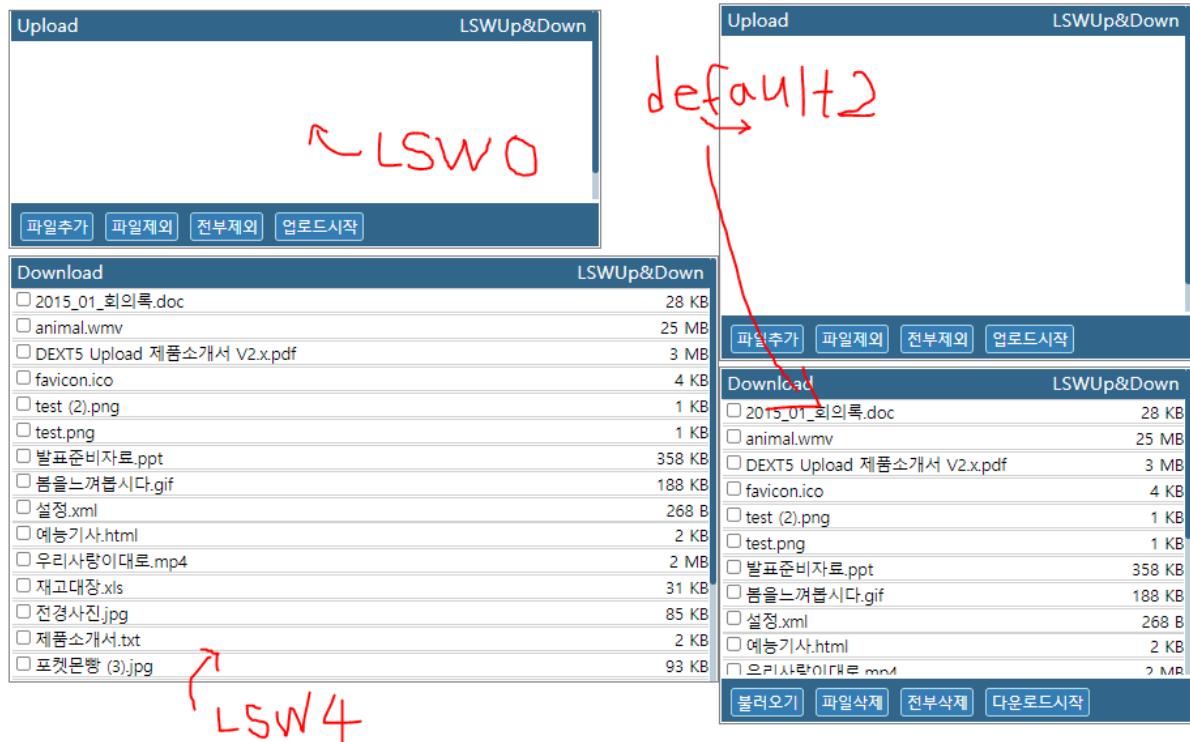
1-1. 생성 예시

```
var LSWup0 = new LSWUpDownPrototype('sample', 'lsw0', 500, 200, 'Upload', 0, loadTest);
// 500px * 200px 의 업로더를 lsw0의 id를 부여하여 버튼을 포함하여 생성 후
// sample 이라는 id의 div에 붙힙니다. 로드가 완료되면 loadTest 라는 함수를 실행합니다.
```

```
var LSWdown2 = new LSWUpDownPrototype('sample', 'lsw4', 600, 400, 'Download', 1);
// 600px * 400px 의 다운로드를 lsw4의 id를 부여하여 버튼을 포함하지않고 창만 생성 후
// sample 이라는 id의 div에 붙힙니다. 로드가 완료되면 기본값으로 설정된 함수를 실행합니다.
```

```
var defaultLSW2 = new LSWUpDownPrototype('sample2', 'default2');
// 기본값의 크기(400px * 300px) 의 업로더와 다운로드를 default2의 id를 부여하여
// 버튼을 포함하여 생성 후 sample2 라는 id의 div에 붙힙니다.
// 로드가 완료되면 기본값으로 설정된 함수를 실행합니다.
```

1-2 생성 예시 결과



객체생성을 통해 간섭없이 독립적으로 기능하는 창을 생성할 수 있습니다.

2. 이벤트 함수 목록

OnUploadDone_LSW : 업로드 완료 시 동작

OnStopUpload_LSW : 업로드 중단 시 동작

OnDownloadDone_LSW : 다운로드 완료 시 동작

OnDeleted_LSW : 서버에서 파일 삭제 완료 시 동작

OnIframeLoaded_LSW : 창이 온로드 되었을 시 동작

총 5개의 이벤트 함수를 구현하였습니다.

2-1. 이벤트 함수 구현 내용 및 기본값

```
LSWUpDownPrototype.prototype.eventList = { // 유저가 사용할 이벤트 함수
  OnUploadDone_LSW : function (result) { // 업로드 완료시
    console.log("Default Message : 업로드가 완료되었습니다.");
    console.log(result); // result[] = 업로드가 완료된 파일 목록
  },

  OnStopUpload_LSW : function (result, uploaded) { // 업로드 중단시
    console.log("Default Message : 업로드가 중단되었습니다.");
    console.log(result); // result[] = 업로드가 중단된 파일의 이름 , 진행률
    console.log(uploaded); // uploaded[] = 이미 업로드가 완료된 파일들
  },

  OnDownloadDone_LSW : function (result) { // 다운로드 완료시
    console.log("Default Message : 다운로드가 완료되었습니다.");
    console.log(result); // result[] = 다운로드가 완료된 파일 목록
  },

  OnError_LSW : function (result) { // 오류 발생시
    console.log("Default Message : 오류가 발생하였습니다.");
    console.log("ErrorCode : "+result); // result = 오류번호
  },

  OnDeleted_LSW : function (result) { // 파일 삭제시
    console.log("Default Message : 파일을 삭제하였습니다.");
    console.log(result); // result[] = 삭제가 완료된 파일 목록
  },

  OnIframeLoaded_LSW : function (Iframe) { // 온로드 되었을때 (사용자가 준함수 or 기본값 사용)
    LSWUpDownPrototype.prototype.option.loaded(Iframe); // Iframe = 생성된 창 의 Iframe객체
  }
}
```

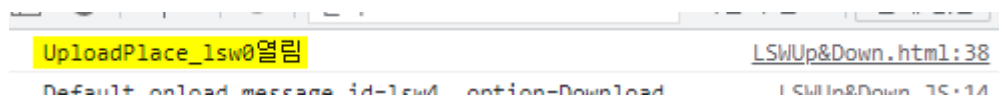
각 상황에 result나 Iframe을 함수의 인자값을 통해 return 해줍니다.

사용자의 소스에서 오버라이드 하여 이벤트 함수로 활용할 수 있습니다.

2-2. 이벤트 함수 활용 예시

```
var loadTest = function(Iframe){
  console.log(Iframe.id+'열림');
  //Iframe.style.background = 'blue';
}
```

온로드 함수를 사용자의 소스에서 생성하여 제품의 객체생성시 파라미터로 줬을 경우



UploadPlace_lsw0열림 LSWUp&Down.html:38

Iframe이 온로드 시 넘겨준 함수를 실행합니다.

```
LSWup0.eventList.OnUploadDone_LSW = function(result){
    console.log("업로드를 다했어요!");
    console.log(result);
}
```

OnUploadDone_LSW를 사용자의 소스에서 재정의 할 경우

업로드를 다했어요! [LSWUp&Down.html:148](#)
[LSWUp&Down.html:149](#)
 ▼ [{...}] ⓘ
 ▶ 0: {name: 'DEXT5 Upload 제품소개서 V2.x.pdf', size: '3 MB', file: File
 length: 1
 ▶ [[Prototype]]: Array(0)

업로드가 완료 시 재정의 한 함수가 실행되며

업로드 된 해당 파일의 정보를 파라미터로 전달해주고 전달받은 파라미터를 활용할 수 있습니다.

```
LSWup0.eventList.OnStopUpload_LSW = function(result,uploaded){
    console.log("업로드 중단했어요!");
    console.log(result);
    console.log(uploaded);
}
```

OnStopUpload_LSW를 사용자의 소스에서 재정의 할 경우

업로드 중단했어요! [LSWUp&Down.html:153](#)
[LSWUp&Down.html:154](#)
 ▼ (2) [{...}, '28.66%'] ⓘ
 ▶ 0: {name: '1GB (10).txt', size: '1 GB', file: F:
 1: "28.66%"
 length: 2
 ▶ [[Prototype]]: Array(0)
[LSWUp&Down.html:155](#)
 ▼ (6) [{...}, {...}, {...}, {...}, {...}, {...}] ⓘ
 ▶ 0: {name: '허니버터칩.png', size: '54 KB', file:
 ▶ 1: {name: '포켓몬뱀 (4).jpg', size: '93 KB', fil
 ▶ 2: {name: 'test (3).png', size: '1 KB', file: F:
 ▶ 3: {name: 'test (2).png', size: '1 KB', file: F:
 ▶ 4: {name: '포켓몬뱀 (3).png', size: '93 KB', fil

업로드가 중단 시 재정의 한 함수가 실행되며

업로드 중이던 파일의 정보와 진행률, 이미 올라간 파일들을 파라미터를 통해 전달해줍니다.

```
LSWdown2.eventList.OnDownloadDone_LSW = function(result){
    console.log("다운로드를 다했어요!");
    console.log(result);
}
```

OnDownloadDone_LSW를 사용자의 소스에서 재정의 할 경우

```
다운로드를 다했어요! LSWUp&Down.html:158
LSWUp&Down.html:159
▶ (3) ['favicon.ico', 'test (2).png', 'test.png']
```

다운로드 완료 시 재정의 한 함수가 실행되며

다운로드한 파일들의 파일명을 파라미터를 통해 전달해줍니다.

```
defaultLSW2.eventList.OnDeleted_LSW = function(result){
    console.log("삭제를 완료했어요!");
    console.log(result);
}
```

OnDeleted_LSW를 사용자의 소스에서 재정의 할 경우

```
삭제를 완료했어요! LSWUp&Down.html:163
LSWUp&Down.html:164
(25) ['1GB.txt', '2015_01_회의록.doc', 'animal.wm
v', 'DEXT5 Upload 제품소개서 V2.x(1).pdf', 'DEXT5
Upload 제품소개서 V2.x.pdf', 'favicon.ico', 'LSWUp
&Down_112.136.138.139, 2022년 04월 08일 15시 39분
..
```

삭제 완료 시 재정의 한 함수가 실행되며

서버에서 삭제된 파일들의 파일명을 파라미터를 통해 전달해줍니다.

3. API 제공

API를 제공하여 기본값으로 설정된 버튼 외에 사용자가 따로 버튼을 만들어 사용이 가능하며 파일명을 받아 파일이 있는지 확인해주거나 해당 파일을 간편하게 다운로드 기능을 제공합니다.

```
LSWUpDownPrototype.prototype.APIList = [ // 유저가 사용할 API
>   LswFileLoadAPI:function (id,fileList){ // 파일 리스트에 올리기 ...
>   },
>   LswFileUpAPI:function (id){ // 리스트에 있는 파일 업로드 ...
>   },
>   LswFileDownAPI:function (id){ // 리스트에 체크한 파일 다운로드 ...
>   },
>   /* LswRefreshAPI() => LswGetPostListAPI() 로 대체 ...
>   LswGetPostListAPI:function (){ // 존재하는 게시물 번호를 전부 출력 ...
>   }
> ]
```

세부 구현 내용과 에러처리는 소스를 참고하시기 바랍니다.

3-1. API 활용 예시

-업로더, 다운로더 생성

```
var LSWup1 = new LSWUpDownPrototype('sample', 'lswUP', 400, 300, 'Upload', 0);
var LSWdown1 = new LSWUpDownPrototype('sample', 'lswDN', 400, 300, 'Download', 0);
```

테스트 페이지에 테스트를 위해 다른아이디의 업로더와 다운로더를 생성해놓았습니다.

-LswFileLoadAPI(id,fileList)

```
var filetag = document.getElementById('fileInputttttt');
filetag.onchange = function(){
    var filelist = filetag.files;

    // 파일 등록 API //
    LSWup1.APIList.LswFileLoadAPI('lswUP',filelist);
    // id, filelist({file1,file2})를 주면 해당 id의 업로드창에 파일들을 올려줌 //
    filetag.value = "";
}
```

-LswFileUpAPI(id)

```

var button = document.querySelector("#uploaddddd");
button.onclick = function(){
    // 파일 업로드 API //
    LSWup1.APIList.LswFileUpAPI('lswUP');
    // id값을 파라미터로 주면 해당 id의 업로드창에 올라간 파일들을 업로드 실행해줌 //
}

```

-LswFileDownAPI(id)

```

var button2 = document.querySelector("#dnloaddddd");
button2.onclick = function(){
    // 파일 다운로드 API //
    LSWdown1.APIList.LswFileDownAPI('lswDN');
    // id값을 파라미터로 주면 해당 id의 다운로드창에 체크된 파일들을 다운로드 실행해줌 //
}

```

-LswGetPostListAPI()

```

var button7 = document.getElementById('listttttt');
button7.onclick = function(){
    // 존재하는 게시물 번호를 전부 출력해주는 API //
    LSWdown1.APIList.LswGetPostListAPI();
    // 웹 콘솔에 현재 존재하는 게시물들의 번호를 전부 출력 //
}

```