UI/UX needs

왜 사용하는가?

왜 이용하는지 / 이용하는데 중요도

서비스 생태계 내에서 탐색/발견과 개입비용이 감소될수록

소비자는 구매결정에 더욱 관대해진다.

사용자 가치 검증으로 제품과 서비스의 UX 품질 완성도 높일 수 있다.

직관적인 아이콘 메타포

시스템 상태 공유 – 피드백 (로그인 유무, 상태정보)

임시 모드 (로그인 서비스, sns계정 공유)

멀티테스킹 – 중요메뉴 접근성

사용자의 실수를 예방하는 장치 마련 – 팝업창 확인, 인터렉션,

심플 데이터 폼

스크롤 인터렉션

히든 툴바 (ptInRect or click)

Swipe to action

Walk throughts

Pull ot refresh

Parallex Scroll

Look & Feel

디스플레이 방식과 사용성 전체의 느낌 시각적 인상

Tone & Manner

그래픽 톤(색감, 질감, 대비대조, 빛) 전달하는 방식

그래픽 인터페이스(GUI) 사용성(Usability)

Navigation – 항해성 > 접근선

Guidance – 학습성 > 효과성

Componenet – 공간효율 > 접근성

Animation – 어포던스 > 인지적 부담감소

Affordance – 행동유도성

Contextual Hovering Menu > 접근성/멀티 이벤트 (누르면 노출)

Carousel

Card UI

디자인 요소 - 레이아웃 배경 색상 톤 빛과 그림자 균형

GUI 무드보드

Look & Feel , Tone & Manner 포함

1. 상단 200px header
2. Content main

* 페이지별 내용, LNB, 배너 게시판 퀵메뉴등
* 해당 페이지에서 나타내고자 하는 내용 작성
* LNB : Local navigation bar – 사이드 메뉴
* 메인메뉴(대메뉴)에 포홤된 하위메뉴(소메뉴)

1. Footer

-회사나 사이트에 대한 정보

-각종 연락처, 관리자 정보 , 약관 및 정책, 카피 라이트등

배경색 여백 테두리

웹앱웹앱 – hybrid

CMS

Word press

Cafe24

Makeshop

Godo mail

Web > hosting

홈페이지 주소 (Domain)

FTP (File Transfer Protocol) – 파일 목록을 볼 수 있는 주소

HTTP (Hyper Text Transfer Protocol) - 파일 내용을 볼 수 있는 주소

Hyper Text : 글뿐만 아니라 그림 영상 등 수정 및 보기 가능

HTTPs (Hyper Text Transfer Protocol Security) – 보안

웹브라우저(edge, chrome) - HTTP 보여주는 프로그램

Html5 – HTTP의 내용을 만드는 언어

PC – FTP - web (hosting)– domain (homepage) – HTTP (html)

<html>

<head>

<title>hello</title>

</head>

<body>

Hello world !!!

</body>

</html>

KWCAG 2.1 = 접근성 검사

웹페이지 개발 방법

분리 코딩

* 구조(structure) : html
* 표현(Presentation) : css
* 동작(Behavior) : javascript

레이아웃 기본 구조

1. header

* 상단 로고, GNB, topmenu 등
* 주로 페이지 안내(이동)에 관한 요소 포함
* GNB : Global Navigation Bar 메뉴바

1. content(main)

* 페이지별 내용, LNB, 배너, 게시판, 퀵메뉴 등
* 해당 페이지에서 나타내고자 하는 내용 작성
* LNB :Local Navigation Bar : 사이드메뉴
* 메인메뉴에 포함된 하위메뉴의 집합

1. footer

* 회사나 사이트에 대한 정보
* 각종 연락처, 관리자 정보, 약관 및 정책, 카피라이트 등
* 기타 요소(SNS, 검색, 관련 사이트 등)는 필요한 위치에 추가

HTML5 Section 요소

**Semantic Elements (의미를 가지는 태그들)**

1. header

* 페이지의 독자적 요소
* 로고, 네비게이션
* 반드시 하나 이상의 제목(h1~h6)명시
* 요소의 위치 무관

1. article

* 페이지 내용상 별개의 내용을 구성할 때 사용되는 섹션

1. section

* 단락, 이미지, 제목 등의 콘텐트를 묶어 더 큰 논리적 단위 형성
* body 안에 들어 갈 수 있는 모든 요소 사용(header, footer 포함)
* class 나 id 속성을 사용하여 섹션에 대한 특징 정보(메타데이터)추가

1. main

* 문서의 주요 내용 정의(요소 내부의 내용은 문서에 고유)
* 문서 내에 한 번만 사용
* article, aside, header, nav, footer, 요소의 하위에 사용 불가
* 사이드 콘텐트, 저작권 정보, 사이트 로고, 검색 양식 등 모든 내용 포함 가능

1. nav

* 페이지 링크, 내부 링크가 있는 섹션 마크업 시 사용
* 메인 네비게이션이 있는 섹션에만 사용

1. aside

* 본문 내용에 대한 추가적인 설명, 부수적 콘텐트 표시에 사용하는 태그

1. footer

* 저작권, 개인정보 보호정책, 연락처 등의 콘텐트
* 문서상의 위치 무관

1. figure

* 이미지에 대한 캡션이나 설명을 위한 태그
* 문서의 흐름에 영향을 주지 않는 콘텐트 마크업에도 사용

URL(Uniform Resourse Location)

1. 파일의 위치

* 절대경로
* 최상위 디렉토리부터 순차적으로 이동하는 경로
* 드라이브명(c:, d)이나 프로토콜(http://, <FTP://)으로시작>
* 기준 파일의 위치에 상관없이 대상 파일의 경로가 동일
* 다른 사이트의 자원을 이용하는 경우
* 상대경로
* 기준 파일과 대상 파일의 위치에 따라 경로 설정
* 동일한 디렉토리에 있는 경우
* 파일명:확장자
* 하위 디렉토리에 있는 경우
* 디렉토리명 / 파일명.확장자
* 상위 디렉토리에 있는 경우
* ../위로 \* 1

../파일명.확장자

복합 사용

최상위 디렉토리 : root

Elements

1. meta

* 문서의 속성 정의
* Head 태그 내에 작성

1. title

* 문서의 제목 정의
* Head 태그 내에 작성

1. h(heading)

* 제목 표시
* h1(큰 제목) ~ h6(작은 제목)
* 줄 바꿈 + 볼드 체

1. p(paragraph)

* 문단 구분

1. br(break)

* 줄 바꿈

1. id

* 고유 속성(변수) 부여 (중복 사용 X)
* name (id와 같은 명령어 지만 중복된 사용 가능)

1. table

* 정보 전달에 활용
* 각항의 셀 개수는 동일
* 같은 행(row)에 있는 셀(cell)들은 높이가 동일
* 같은 열(columm)에 있는 셀(cell)들은 높이가 동일

관련태그

* tr요소 : row (횡렬의 셀)로 정의
* th요소 : column (종렬의 셀)으로 제목 셀을 정의
* td요소 : cell (표 안의 항목)로 데이터 셀을 정의
* col요소 : 열의 속성값과 스타일을 위한 그룹화
* colgroup요소 : 열의 구조적 그룹화. col요소를 포함할 수 있다.
* colspan : 열 병합
* rowspan : 행 병합

병합할려는 셀의 시작부분에 작성

집근성 향상을 위한 요소

* caption요소 : 표의 제목을 정의
* thead요소 : 헤더 행을 그룹화하여 정의
* tfoot요소 : 푸터 행을 그룹화하여 정의
* tbody요소 : 본체 행을 그룹화하여 정의
* summary(속성) : 표의 제목
* id(속성) : 제목셀 셀에 이름 부여
* headers(속성) : 해당 셀의 제목셀을 지정

1. <input type="종류 선언”>

* type : 필수속성입니다. 다음은 타입속성에서 사용할 수 있는 값입니다.
* text : 단일행 텍스트 입력필드
* password : 비밀번호 입력필드. 감춰져서 출력된다.
* checkbox : 체크박스 생성. 복수 선택이 가능 (content : checked 체크된상태)
* radio : 라디오버튼 생성. 복수 선택 불가능 (name 값이 같아야 1개만 선택)
* submit : 송신버튼 생성 (사용자가입력한 데이터를 action 값으로 전송)
* reset : 리셋버튼 생성
* button : 범용버튼 생성.
* image : 이미지버튼 생성.
* file : 송신파일 선택필드 생성
* hidden : 감춰진 필드.( 이전 페이지 정보 담아두기 ,로그인 후 이전 페이지로 리턴)
* placehoder : input 박스안에 텍스트 생성 및 유지
* value: input 박스안에 텍스트 생성 하지만 유지 X
* name="이름"(필수 속성) : 컨트롤의 이름 지정. type의 속성이 checkbox나 radio일 경우 name속성이 같은 값으로 지정되면 하나의 그룹으로 취급한다.
* value="값" : 컨트롤 종류에 따른 초기값 지정 **(체크 or 입력 값 전송)**
* size="폭" : 컨트롤의 폭을 지정. type속성이 text이거나 password일 경우 문자수를 지정하고 그외의 경우에는 가로폭의 픽셀값을 지정한다.
* width : 요소의 가로사이즈를 지정합니다.
* height : 요소의 높이를 지정합니다.
* max : input요소에 허용하는 값의 최대값을 지정합니다.
* maxlength="최대 입력 문자수" : type속성이 text이거나 password일 경우 최대 입력 문자수를 지정한다.
* min : input요소에 허용하는 값의 최소값을 지정합니다.
* readonly : 쓰기 X 전송 O
* disable : 쓰기 X 전송 X

1. select : 콤보 상자 작성

* <select>

<option>~</option>

<option>~</option>

</select>

<textarea> : 텍스트를 여러 줄 입력하는 상자 작성

<button> : 버튼 형식의 요소 작성(input 대신 button 사용)

* <button type=””>TEXT</button>

1. fieldset : 양식의 소그룹
2. legend : fieldset의 제목
3. label : 필드와 텍스트의 그룹

* label 하나는 하나의 input만 사용한다.

1. for(속성) : 연결하고자 하는 필드의 id 값 작성

ex) <label for="id값”></label>

1. form

* 모든 양식은 <form> ~ </form> 안에서 작성
* 모든 양식은 이름(name AND id) 입력
* 대부분의 양식은 <input> 태그의 “type” 속성으로 정의

<form name="” action="” method="">

<form name="폼 요소의 이름” action="데이터를 처리할 페이지 경로” method="데이터 전송 방식">

데이터 전송 방식

get : 빠름 /보안성 없음

post : 느림 / 보안성 있음

블록레벨(Block Level) – 텍스트 중간에 태그 사용 시 줄이 바뀌는 명령어들

* 독립된 행을 갖는 요소 집합
* 크기 지정, 위치 조정 가능
* p, h1~h6, ul, ol, li, div, table.
* 다른 블록 레벨 요소와 인라인 요소 포함 가능
* p, h, dt, address 같은 일부 요소는 다른 불록 포함 불가

인라인 레벨(Inline Level

* 범위 표시 요소의 집합
* 크기(예외:img) 위치 조정 불가
* a, em, strong, img, span…
* 블록 레벨 요소는 포함할수 없고 인라인 요소만 포함 가능
* html5 이후 a 요소는 블럭 포함 가능
* 높이 속성 사용 불가

CSS(Cascading Style Sheet) lv3

웹 페이지의 표현(디자인)을 나타내는 언어

selector { property : value } > float >

우선순위

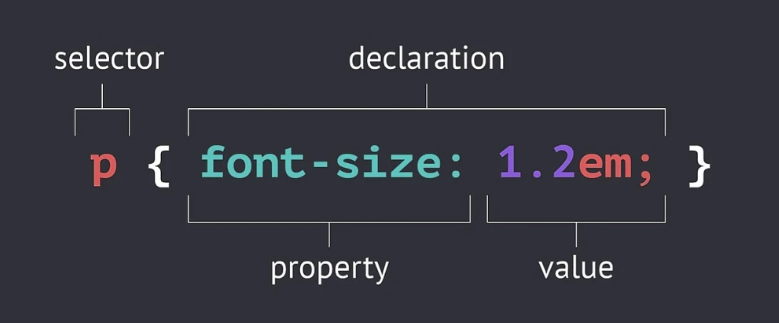
p : A / F > float > m : A (+width, p : R)

**p : Positon m : margin**

**A : Absolute A : auto**

**F : fixed p : positon**

**r : relative**



< A B = "C” B = "C” >

selector(선택자: 누구한테) {

property(속성 : 어떤 효과) : value (값:얼만큼);

}

1. 선택자

* html 태그에 직접 효과를 적용
* 해당 태그 작성 시 자동으로 효과 적용

1. 클래스 선택자

* 중복 가능
* .name { 효과 }

<요소 class="name”>

1. 아이디 선택자

* 중복 불가 (한번만 사용 및 불러오기 가능)

#name { 효과 }

<요소 id="name”>

1. 다중 선택자

* A, B, C { 효과 }

1. 후손 요소 선택자

* A B { 효과 } 부모 o 후손 o

if( A태그 안에 B가 있으면 ture)

1. 자식 요소 선택자

* A > B { 효과 } 부모o 후손 x

스타일 적용방법

* 내부 스타일

<head>

<style type=”text”> 🡨 스타일 선언문

selector { property : value }

</style>

* 인라인 방식 (잘안씀)

<elem style=”property:vaoue, property:value”>

* 외부 방식 (자주씀)

selector { property : value } 🡨 name.css 로 저장 선언문 생략

* 링크 방식 (자주씀)

<head>

<link rel=”stylesheet” type=”text/css” href=”name.css”

* 임포트 방식 (잘안씀)

<head>

<style type=”text”>

@import “name.css”;

</style>

overflow : 블록안에 넘치는 부분을 제어

scroll, visible, hidden, overlay

display-태그가 보이는 방법의 정의함 ex) 블록레벨>인라인레벨로

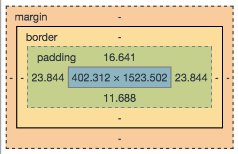
block/inline/inline-block/none

visibility – 보이는 효과

visible/hidden

레이아웃 속성

여백 속성



1. margin

* 사이즈 조절 시 위치가 변함

1. padding

* 사이즈 조절 시 크기가 변함

1. text-align : center

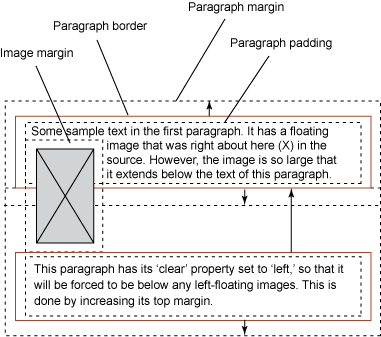
* 인리안 요소 가운데 정렬

1. box-sizing

box-sizing 속성은 CSS의 테두리 영역의 크기를 결정합니다. box-sizing 속성값에는 content-box와 border-box가 있습니다.

* **content-box** : 지정한 CSS width 및 height를 컨텐츠 영역에만 적용합니다. border, padding, margin은 따로 계산되어 전체 영역이 설정값보다 커질 수 있습니다.
* **border-box** : 지정한 CSS width 및 height를 전체 영역에 적용합니다. border, padding, margin을 모두 합산하기 때문에 컨텐츠 영역이 설정값보다 적어질 수 있습니다.

1. float



* 수직 구조의 요소를 수평 구조로 변경
* 요소의 레벨이 inline-block 형식으로 변경
* 수평으로 나열 될 모든 요소에 float 속성 적용
* none/left/right
* 선택된 요소를 위로 띄우는 속성
* 위로 이동하면서 생긴 빈 자리에 다른 요소가 이동
* 자식 요소의 길이 합이 부모 요소의 길이보다 길면 부모 요소의 길이를 초과하는 요소들이 아래쪽으로 이동 (태그 작성순)
* 형제와 부모 관계의 요소들끼리만 적용
* **부모 요소는 float이 적용된 자식 요소의 높이 인식 못함**
  + **자식이 float 일때 부모 height 값 적용 / overflow : hidden**

1. positon ( move 방법?)

* 움직이는 방법
* 위치 속성과 함께 사용

top / right / bottom / left

positon 속성이 없는 경우 이동 불가

right / bottom은 relative인 경우 사용 불가

* 속성값
* static : 기본값, 이동 불가
* absolute :절대 위치 지정, 형제 요소 간 동일한 중싱점 사용
* relative : 모든 요소가 독립된 중심점 사용
* fixed : 스크린을 기준으로 지정된 위치에 고정
* **자식 position 요소가 absolute 일 때, 부모의 positon 요소는 relative 쓰기**

\*\* IR (image Replacement)

* 텍스트를 이미지로 변환
* 적용할 태그에 css 먼저 적용(width , height)

css

        .logo {

            width: 184px;

            height: 39px;

            border: 1px solid black;

        }

        .logo a {

            /\* 인라인 일 때 display:block 적용 \*/

            display: block;

            /\* 로고사이즈 적용 \*/

            width: 184px;

            height: 39px;

            /\* 이미지 불러오기 / 반복 적용 x \*/

            background: url(images/logo.jpg) no-repeat;

            /\* 텍스트 disable/ 들여쓰기 \*/

            text-indent: -9999px;

            overflow: hidden;

        }

적용

        <h1 class="logo"><a href="#">한진</a></h1>

1. Flex(Flex Box, Flexbox) 속성

* 컨테이너에 적용하는 속성
* 아이템에 적용하는 속성
* display:flex;

.container {

display:flex

/\* display:inline-flex\*/

* 아이템들이 가로 방향으로 배치
* 내용물 만큼 width 적용 (inline)
* 아이템들의 height가 모두 컨테이너의 높이만큼으로 자동 적용

(float은 각각의 콘텐트만큼 높이 설정)

* Flex Container(플렉스 컨테이너)
* Flex의 영향을 받는 전체 공간

Flex Item(플렉스 아이템)

* 설정된 속성에 따라 각각의 아이템들이 어떤 형태로 배치되는 것

<div class=”container>

<div class=”item”>helloflex</div>

<div class=”item”>abc</div>

<div class=”item”>dddaw2e423423</div>

</div>

속성

* flex-direction : 배치 방향 설정
* row : 왼쪽부터 순차적 배치
* row-reverse : 오른쪽부터 순차적 배치
* column : 위에서 아래로 순차적 배치
* column-reverse : 아래에서 위로 순차적 배치
* flex-wrap : 줄 넘김 처리 설정 - 아이템 줄 바꿈 속성
* nowrap : 줄 바꿈을 하지 않음
* wrap : 줄 바꿈
* wrap-reverse : 역순으로 배치 후 줄 바꿈
* flex-flow
* flex-direction과 flex-wrap을 한번에 지정
* flex-direction, flex-wrap의 순으로 공백을 구분

flex-flow : direction값 wrap값

ex) flex-flow : row wrap;

* justify : 수평축 정렬
* justify-content : 메인 축 방향 정렬
* 수평축 방향으로 아이템 정렬

justify

* align-items : 수직 축 정렬
* 수직 축 방향으로 아이템 정렬
* align-content : 여러 행 정렬
* flex-wrap : wrap이 설정된 상태에서 아이템들의 행이 2줄 이상인 경우 수직축 방향 정렬 지정
* align-self
* align-items로 교차 축(cross axis)아이템 정렬
* align-items 속성보다 우선 적용
* 컨테이너 안의 모든 아이템에 적용
* 특정 아이템의 정렬을 따로 정할 경우 align-self 속성사용
* 기본값은 auto align-items 속성의 값을 상속
* auto : 부모 컨테이너의 align-items 속성을 상속받음.
* 부모 컨테이너가 없는 경우 stretch 적용
* stretch : 컨테이너에 맞게 변경
* center : 컨테이너 중심에 위치
* flex-start : 컨테이너 앞단 부분에 위치
* flex-end : 컨테이너 끝단 부분에 위치
* baseline : 컨테이너의 기본선에 위치

        .justify-content {

            /\* 기본값 아이템들을 시작점으로 정렬 - flex direction : row(왼쪽) column(위)\*/

            justify-content: flex-start;

            /\* 아이템들을 끝점으로 정렬 - flex direction : row(왼쪽) column(위)\*/

            justify-content: flex-end;

            /\* 아이템들을 가운데로 정렬 \*/

            justify-content: center;

            /\* 아이템들 사이에 균일한 간격 생성 ㅁ(간격)ㅁ(간격)ㅁ(간격)ㅁ \*/

            justify-content: space-between;

            /\* 아이템의 좌우에 균일한 간격 생성 (간격)ㅁ(간격)ㅁ(간격)ㅁ(간격)ㅁ(간격)\*/

            justify-content: space-around;

            /\* 아이템들의 사이와 양 끝에 균일한 간격 생성(좌우 맨끝도 같게) \*/

            justify-content: space-evenly;

        }

        .align-content {

            /\*  justify-content 동일 ->수직 \*/

            align-content: flex-start;

            align-content: flex-end;

            align-content: center;

            align-content: space-between;

            align-content: space-around;

        }