#1.0 Why we need Flexbox

* + 복잡한 계산 없이 박스의 크기와 순서를 배치할 수 있음
  + 정렬, 방향, 순서, 사이즈 등을 유연하게 조절하기 때문에 편리

#1.1 Basics of Flexbox

* + Flex의 기본 개념 두 가지는 Flex-container와 Flex-item
  + 문서의 영역 중 flexbox가 놓여있는 영역을 Flex-Container라고 함
  + Flex-container와 Flex-item는 직접적으로 종속관계에 있어야 함
  + Flex-item에게 규칙을 주는 게 아니라 Flex-container에게 명령을 내림
  + father에게 명령한 규칙은 children items에게 영향을 끼침
    1. space-between : 아이템을 일정한 간격으로 떨어뜨려 정렬
    2. flex-start : 아이템을 모두 앞 쪽에 정렬 (flex를 주면 flex-start가 디폴트)
    3. flex-end : 아이템을 모두 끝 쪽에 정렬
    4. space-around : 아이템 주변에 간격을 만들어 줌

#1.2 Main Axis and Cross Axis

* + align-items은 수평적으로 정렬되어 있는 item을 수직 정렬해 줌
  + 수직 정렬을 위해선 충분한 width(너비)와 height(높이)를 가져야 함
  + 수직, 수평 정렬에 있어 밑바탕이 되는 규칙을 axis(축)이라 함
  + main axis는 justify-content가 동작하는 부분
  + 즉, main axis는 justify-content가 영향을 주는 축이고, cross axis는 align-items가 영향을 주는 축임
  + main axis에 있는 아이템을 움직이려면 justify-content, cross axis에 있는 아이템을 움직이려면 align-items
  + flex 옵션을 주면 기본적으로 flex-direction: row라는 옵션을 가지게 됨 (row를 column으로 변경하면 세로로 정렬)
  + flex-direction이 row면 main axis는 수평, cross axis는 수직이 됨
* flexbox direction이 row면, main(수평) / cross(수직)
* flexbox direction이 column이면, main(수직) / cross(수평)
  + align-items
    1. flex-start : item을 모두 위쪽에 정렬
    2. flex-end : item을 모두 아래쪽에 정렬

#1.3 Flex Wrap and Direction

* + 아이템들끼리 공간이 부족할 때 flex는 기본적으로 찌그러지는 현상이 발생, 이 때 flex-wrap를 사용함
  + flex-wrap: wrap; 을 사용하 면, 아이템들이 겹치는 현상을 방지할 수 있음
  + flex-direction
    1. row-reverse : 정렬 순서를 반대로 바꿔 줌 (ex : 1, 2, 3 => 3, 2, 1)

#1.4 Align Self and Flexbox Conclusions

* + align-self은 아이템들에게 직접 옵션을 주는 속성

#2.1 Basics CSS Grid

* + Grid에는 rows, column이라는 개념이 있음 – 테이블과 같은 개념
  + 그리드 컨테이너에서는 rows, columns를 정의할 수 있음
  + father에는 display: grid, column, row, gap이라는 개념이 있고 이는 자동으로 아이템을 정렬해줌

#2.2 Auto Rows and Columns

* + grid는 초과로 존재하는 children이 있을 때 row/column에 놓을지를 결정하는 속성이 있음
  + 기본적으로 grid-auto-flow는 flexbox와 마찬가지로 row임
  + 이는 row/column으로 수직/수평 정렬을 정의할 수 있음

#2.3 Grid Template Areas

* + Grid template areas를 사용하면 아이템의 위치를 직접 계산하지 않아도 자동 배치됨
  + 화면에 공간을 그린 후 안을 채우는 것과 같음
  + 레이아웃을 만들 때, header, sidebar, content, Footer 네 개의 주요 영역으로 구분할 수 있음
  + 셀을 공백으로 남겨두려면 마침표(.)를 사용하면 됨

#2.4 fr and repeat()

* + Fr 단위는 그리드 컨테이너에 남아있는 사용 가능한 공간의 일정 비율을 나타냄
  + 남아있는 공간에 따라 확장/축소되는 트랙을 생성함
    1. grid-template-columns: 2fr 1fr 1fr;

|  |  |  |
| --- | --- | --- |
| **2fr** | **1fr** | **1fr** |

* + 1. grid-template-columns: 500px 1fr 2fr;

|  |  |  |
| --- | --- | --- |
| **500px** | **1fr** | **2fr** |

* + - 1. 픽셀 값이 지정된 트랙을 제외하고 남은 공간을 나누어 할당
  + Repeat() 표기법을 사용하여 트랙의 전체 또는 일부분을 반복해서 나열해줄 수 있음
    1. grid-template-columns: **1fr 1fr 1fr;** => grid-template-columns: **repeat(3, 1fr);**

|  |  |  |
| --- | --- | --- |
| **1fr** | **1fr** | **1fr** |

* + 1. grid-template-columns**: 20px repeat(2, 1fr) 20px**;

|  |  |  |  |
| --- | --- | --- | --- |
| **20px** | **1fr** | **1fr** | **20px** |

* + - 1. 20 픽셀의 트랙을 생성하고, 1fr 크기의 트랙을 2번 반복해서 채운 후 마지막에 20픽셀 트랙을 생성
    1. grid-template-columns: **repeat(2, 1fr 2fr);**

|  |  |  |  |
| --- | --- | --- | --- |
| **2fr** | **1fr** | **2fr** | **1fr** |

* + - 1. 2fr + 1fr 트랙이 두 번 반복

#2.5 minmax, max-content, min-content

* + minmax : 오브젝트 크기의 최소/최대값을 지정할 수 있음
  + max-content : 내용의 크기(길이)에 맞춰 공간을 할당 – title같은 걸 만들 때 유용
  + min-content : 가질 수 있는 가장 적은 공간을 할당
  + 복잡한 과정 없이 간단하게 content를 자유자재로 다룰 수 있음

#2.6 auto-fill, auto-fit

* + auto-fill : 가능한 많은 cell로 container를 채워 줌
  + auto-fit : 가능한 많은 content를 받아와서 펼쳐 줌

#2.7 Justify Content, Align Content and Place Content

* + justify-content는 모든 column을 통째로 옮김 – grid 자체를 움직이는 것 (가로)
  + align-content는 모든 box가 포함되어 있는 grid container를 이동 (세로)
  + place-content
    - 1. justify-content와 align-content를 한 번에 실행할 수 있음
      2. place-content : align값 justify값

#2.8 Justify Items, Align Items and Place Items

* + justify-content와 align-content는 box 자체를 상,하,좌,우로 움직이는 방법이라면
  + justify-items와 align-items는 box 안에 있는 content를 움직이는 방법임
  + place-items를 사용하면 justify-items와 align-items를 한 번에 실행할 수 있음

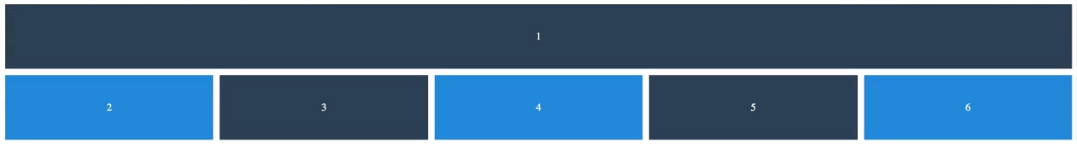
#2.9 Grid Column, Row Start and End

* + grid column은 column을 병합해줌 – line을 기준으로 잡음
  + grid column은 father 쪽이 아닌 items에서 작성해야 함
    1. grid-column : 1 / 3;



* + 1. grid- column-start : 1;

grid- column-end : -1;



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **line** | **1** | **2** | **3** | **4** | **5** | **6 or -1** |

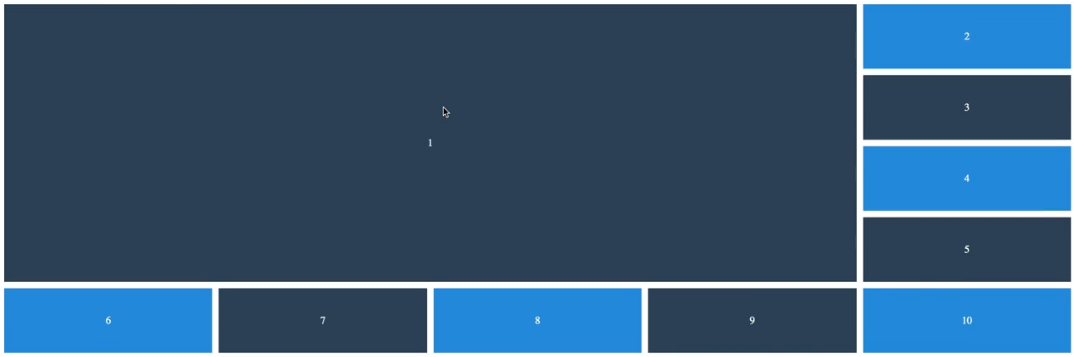
#2.10 Line Naming and grid-auto-flow

* + line naming을 할 땐 배열[]로 저장해야 하며 빈칸이 있으면 안됨 (‘-‘로 연결되어 있어야 함)
  + grid-auto-flow: row dense; => 모든 빈칸을 채워줌

#2.11 Grid Row, Row Start and End

* + grid-column : span 4; – 4칸의 column을 병합 (span은 line 기준이 아닌 셀 기준)

grid-row : span 4; – 4칸의 row를 병합



#2.12 Grid Area

* + grid area는 span과 비슷한 기능
  + grid-area: row start, column start, row end, column end 순으로 값을 주면 됨
  + 값을 두 개만 넣으면 row, column의 값으로 생각함

#2.13 Justify, Align, Place Self

* + 위에서 배웠던 justify는 전체 아이템에 대해서 적용했지만,
  + place-self는 하나의 아이템 대해서만 적용됨
    - 1. place-self : align값 justify값;

PostCSS / CSSNext / CSS4

#3.0 Installing Parcel