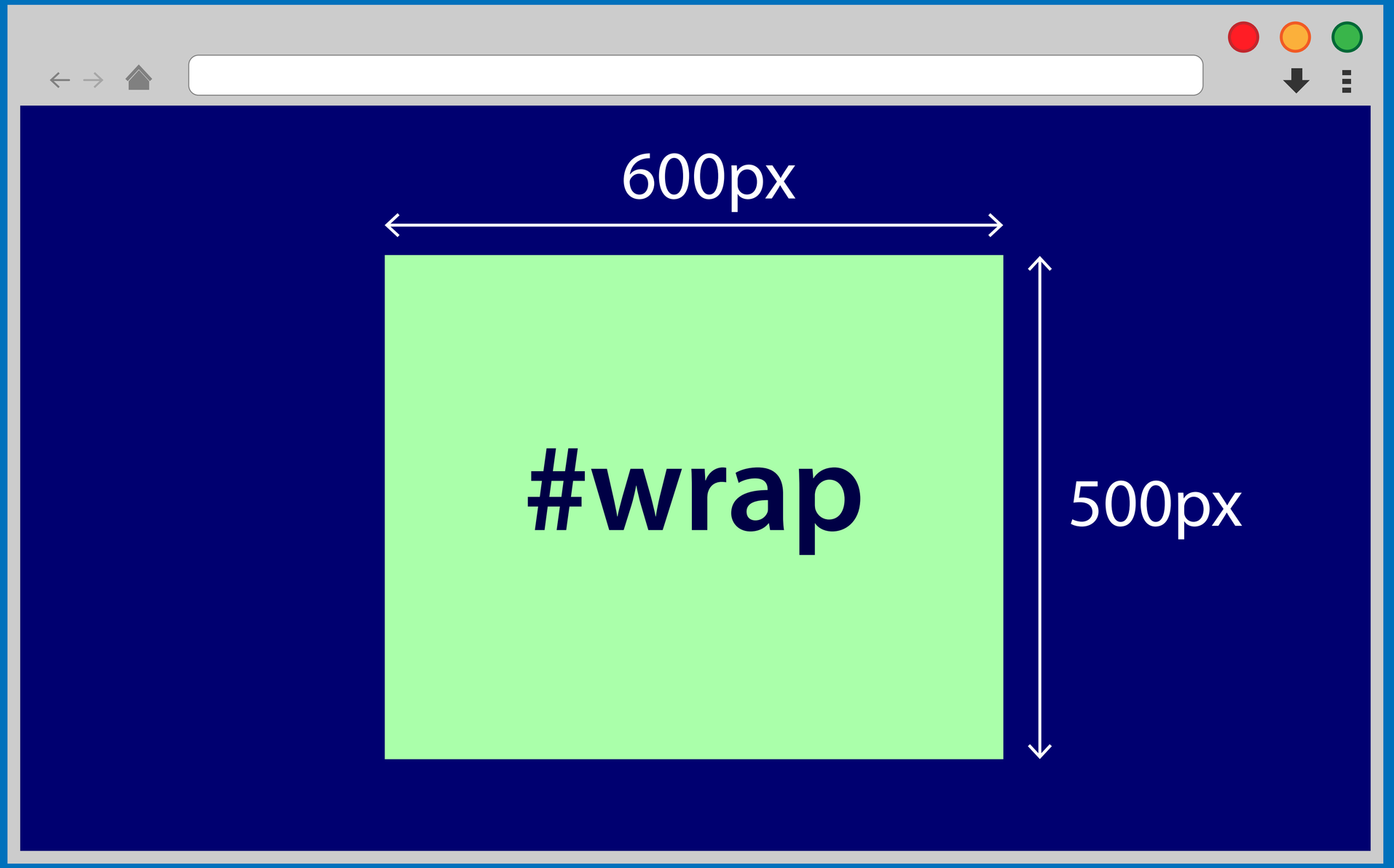
WEB LAYOUT
BASIC : 001

사전 제작물 점검하기

layout - 01

완성 결과물 보고 제작하기

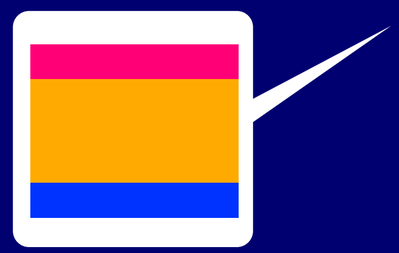


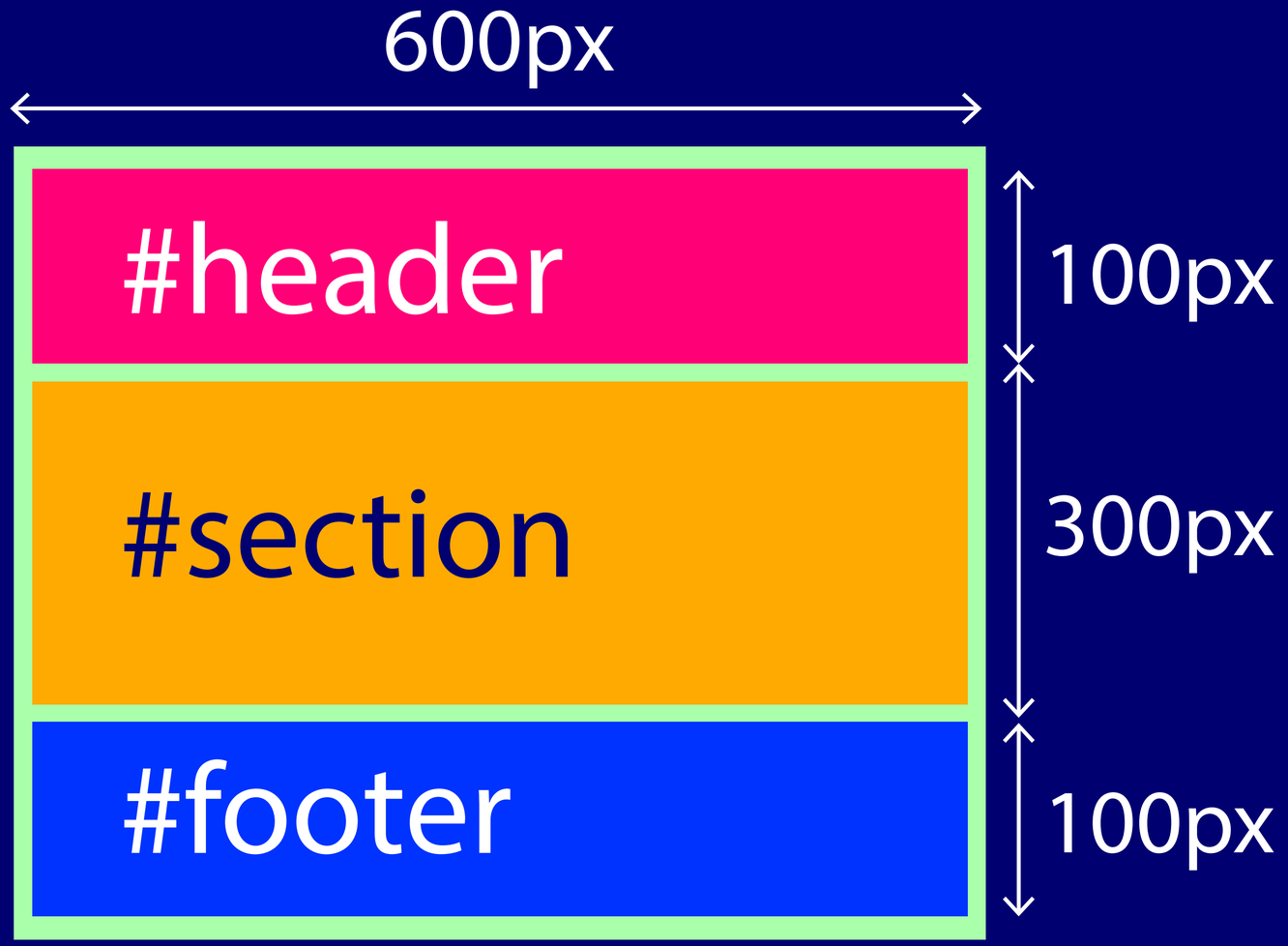
#000070

#AAFFAA

layout - 02

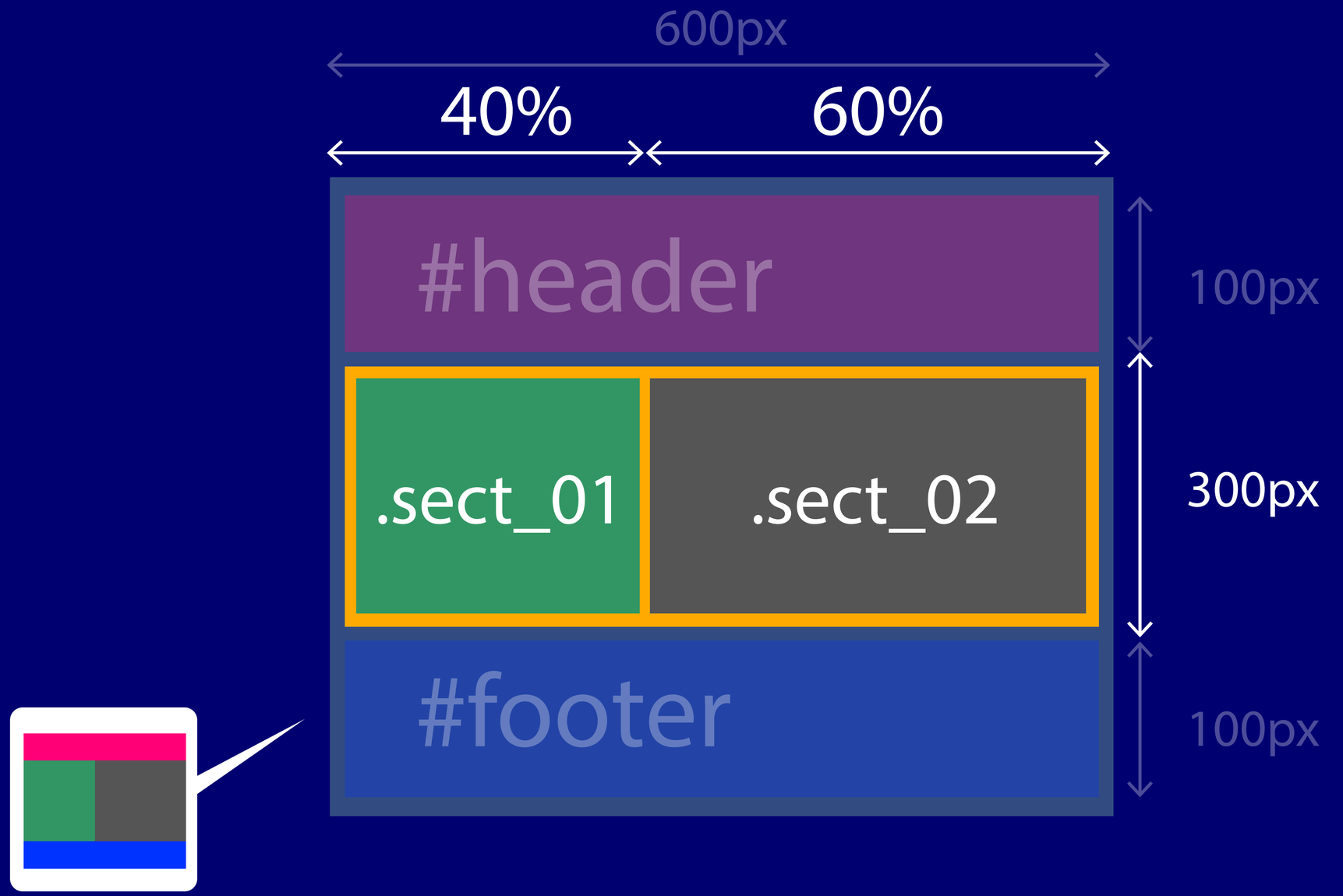
#wrap 내부 만들기





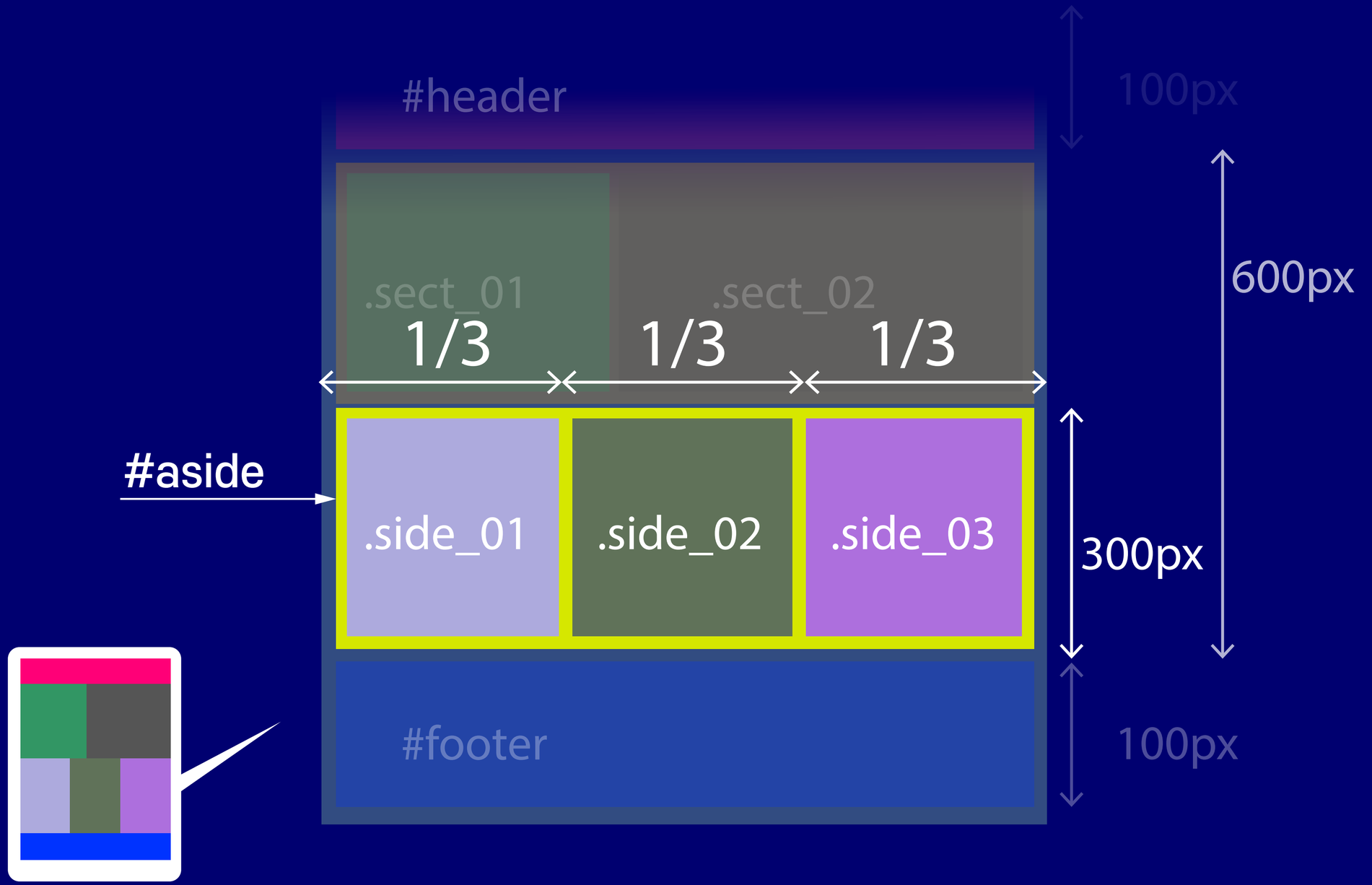
layout - 03

#section 내부 만들기



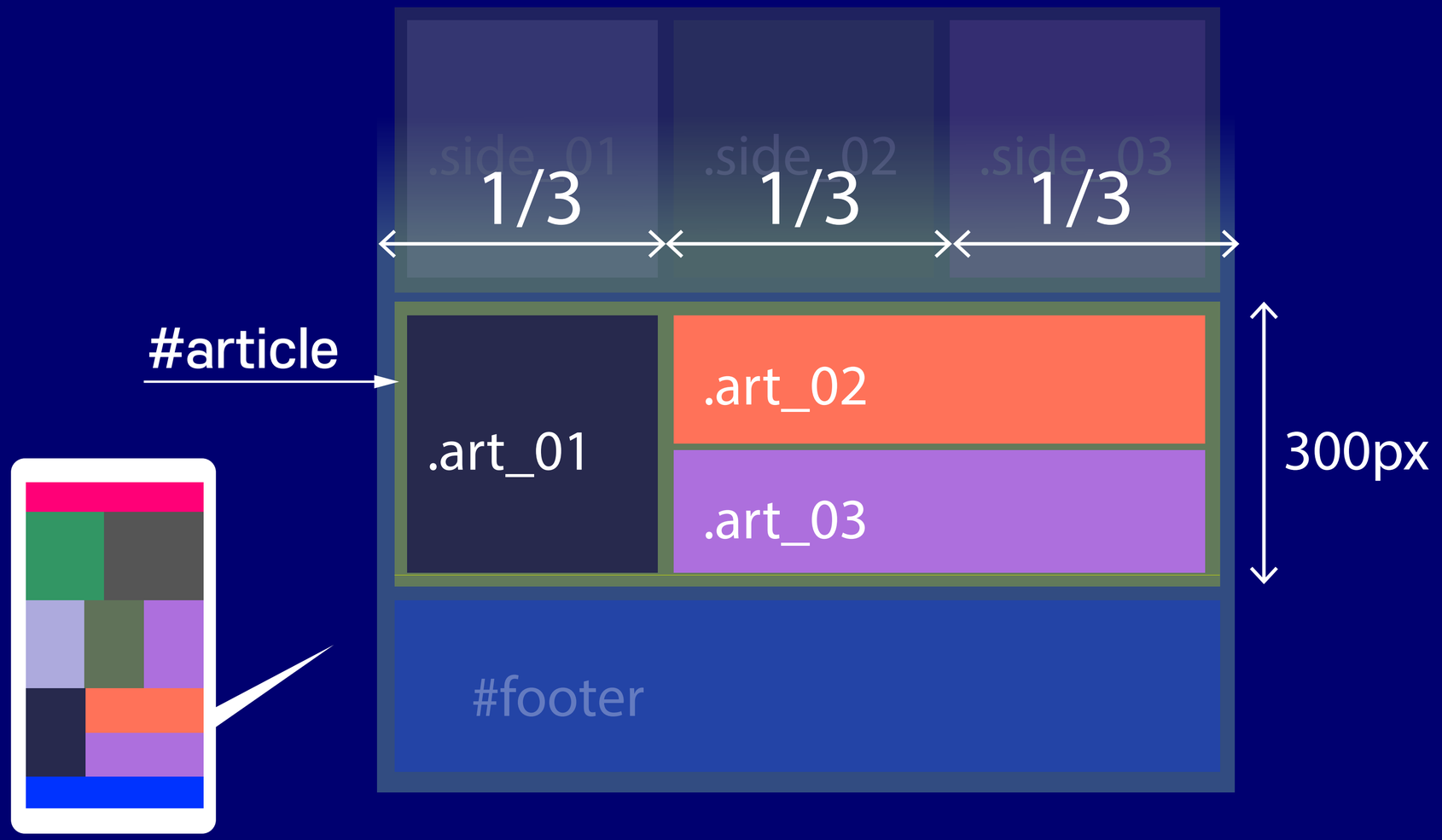
layout - 04

#section 뒤 추가 만들기



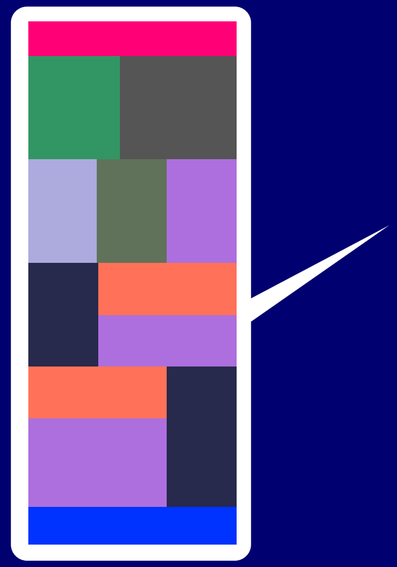
layout - 05-1

#aside 뒤 추가 만들기



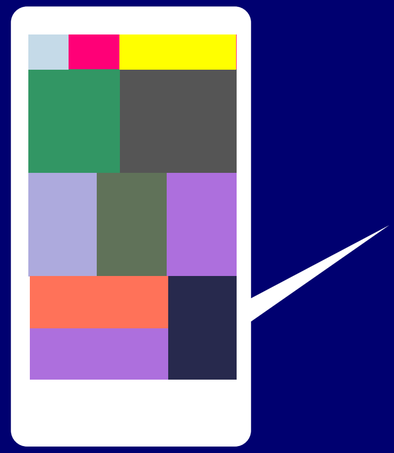
layout - 05-2

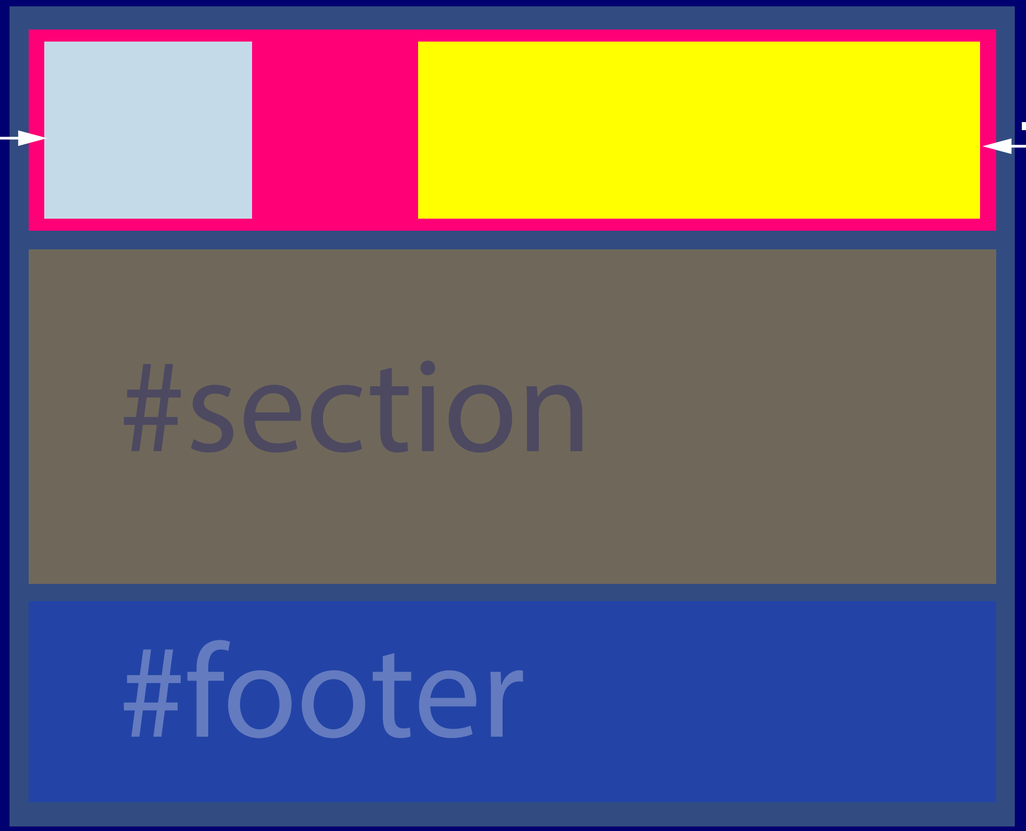
#article 뒤 추가 변경하기



layout - 06

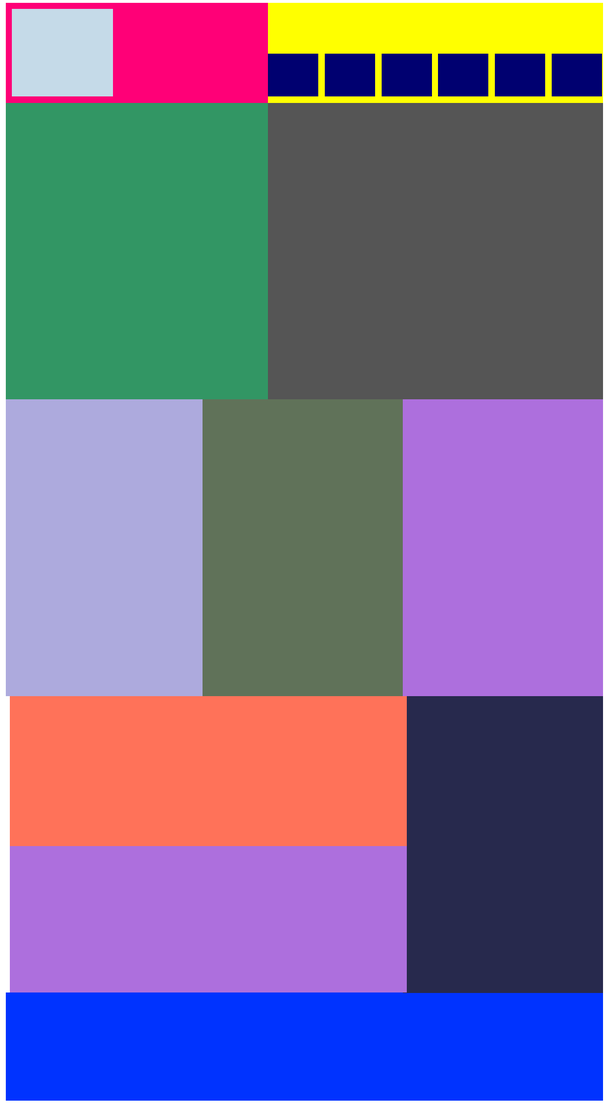
#header 내용 형태 변경하기





layout - 07

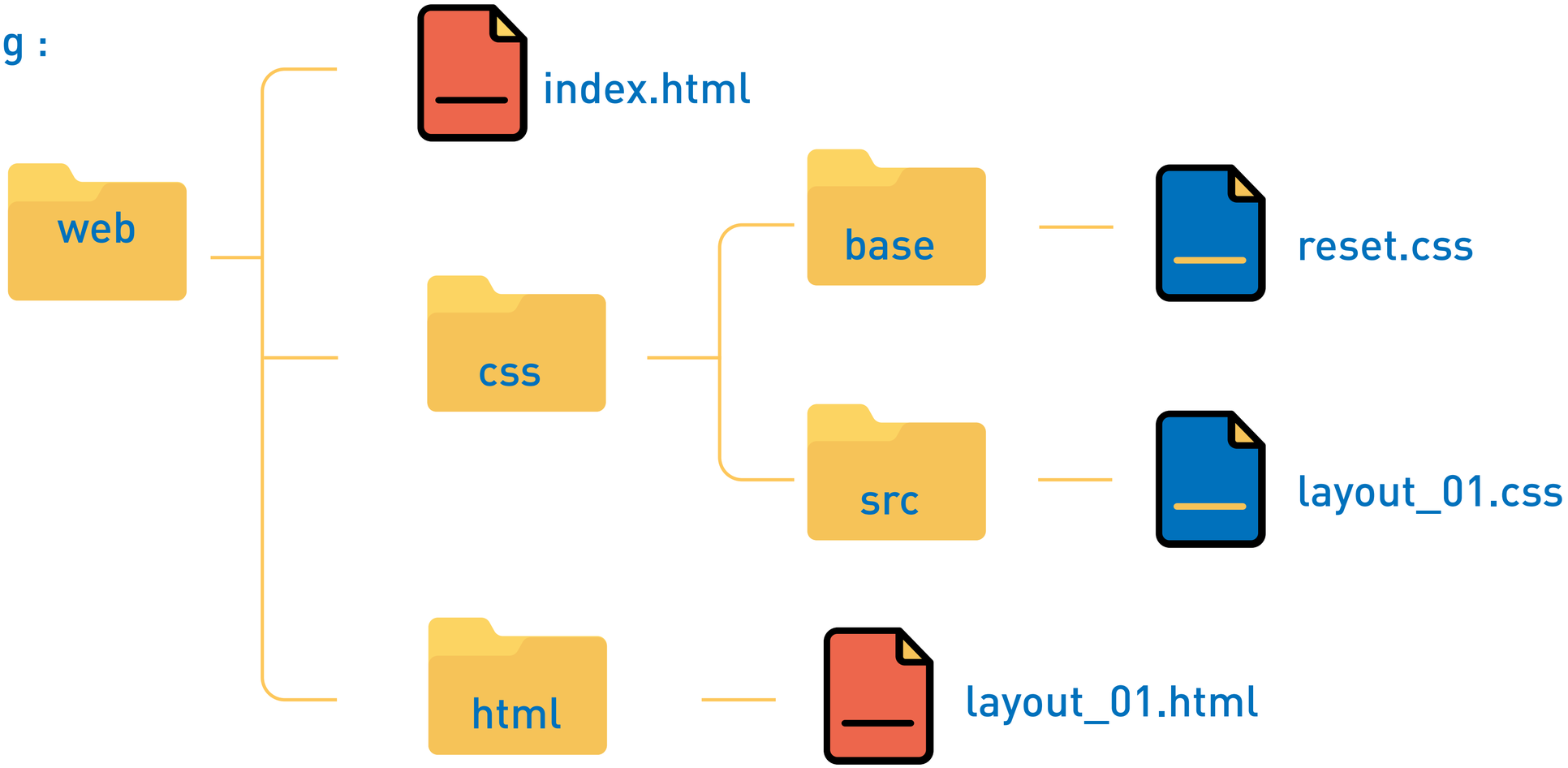
최종형태 파악하기



세팅 및 레이아웃 제작하기

STEP_01

Basic Setting : file tree



STEP_01
index.html

```
<!DOCTYPE html>
<html lang="ko-KR">
<head>
  <meta charset="UTF-8">
  <title>web layout</title>
</head>
<body>
  <script>
    // index.html 접속시
    // "./html/layout_01.html" 파일위치로 강제 이동 처리하는 script 명령어
     window.location = "./html/layout_01.html";
  </script>
</body>
</html>
```

STEP_01
layout_01.html

```
<!DOCTYPE html>
<!-- layout_01.html -->
<html lang="ko-KR">
<head>
  <meta charset="UTF-8">
  <link rel="stylesheet" href="../css/src/layout_01.css">
  <title>web layout</title>
</head>
<body>
  <!-- layout 구조 작업 -->
  <!-- code 작업 내용부분 -->
  <h1>layout 01</h1>
  <div class="test">test</div>
  <!-- layout 구조 끝 -->
</body>
</html>
```

css파일을 외부에서 작성하여,
불러오는 방식
(html문서기준으로css파일 찾아오기)

STEP_01
layout_01.css

```
/* layout_01.css */

h1{
  width:800px; heigh:100px;
  backround-color:#acf;
}

.test {
        width:500px; height:200px;
        background-color:#ccc;
}
```

reset.css 기능 제작
(하나의 파일에 초기화 처리 후
  차후 별도의 페이지로 분류)

1. 결과물이, 웹페이지의 끝에 나타나지 않고 약간의 공백을 가지고 있음
2. F12를 이용하여 각 코드의 공백이 무엇을 의미하는지 파악!
3. 웹내부의 코드는 모두 각자 필요한 여백,선,색상 등의 다양한 기능들을 가지고 있으며,
   브라우저/앱 등 각자 고유한 값을 별도로 가지고 있으므로 이를 초기화 시킬 필요가 있음

STEP_01
layout_01.html

기존 내용
삭제 후

```
<!DOCTYPE html>
<!-- layout_01.html -->
<html lang="ko-KR">
<head>
  <meta charset="UTF-8">
  <link rel="stylesheet" href="../css/base/reset.css">
  <link rel="stylesheet" href="../css/src/layout_01.css">
  <title>web layout</title>
</head>
<body>
  <!-- layout 구조 작업 -->
  <!-- code 작업 내용부분 -->
  <h1>layout 01</h1>
  <div class="test">test</div>
  <!-- layout 구조 끝 -->
</body>
</html>
```

css파일을 기능별로 구분하여,
각각 필요한 내용을 외부에서 불러오기
(css 문서의순서를 명확하게 처리해야함)

STEP_01
reset.css

```
/* reset.css 영역 ------------------------------ */
* {
  margin:0; padding:0; border:0;
  font-family:sans-serif;
}

hr {
  width:100%; height:0;
  border-bottom:2px solid #333;
}
```

1. 실제 적용시 “*” 표기를 이용한 선택자는 사용하거나 권장하지 않음
2. 별도의 reset.css 파일을 생성하여 처리하는것이 권장하는 형태
3. normalize.css , meyer’s reset.css , naver reset.css , daum reset.css 등을 참조하여 위 내용을 명확하게 구분하여 채워볼것!

STEP_01
layout_01.css

기존 내용
삭제 후

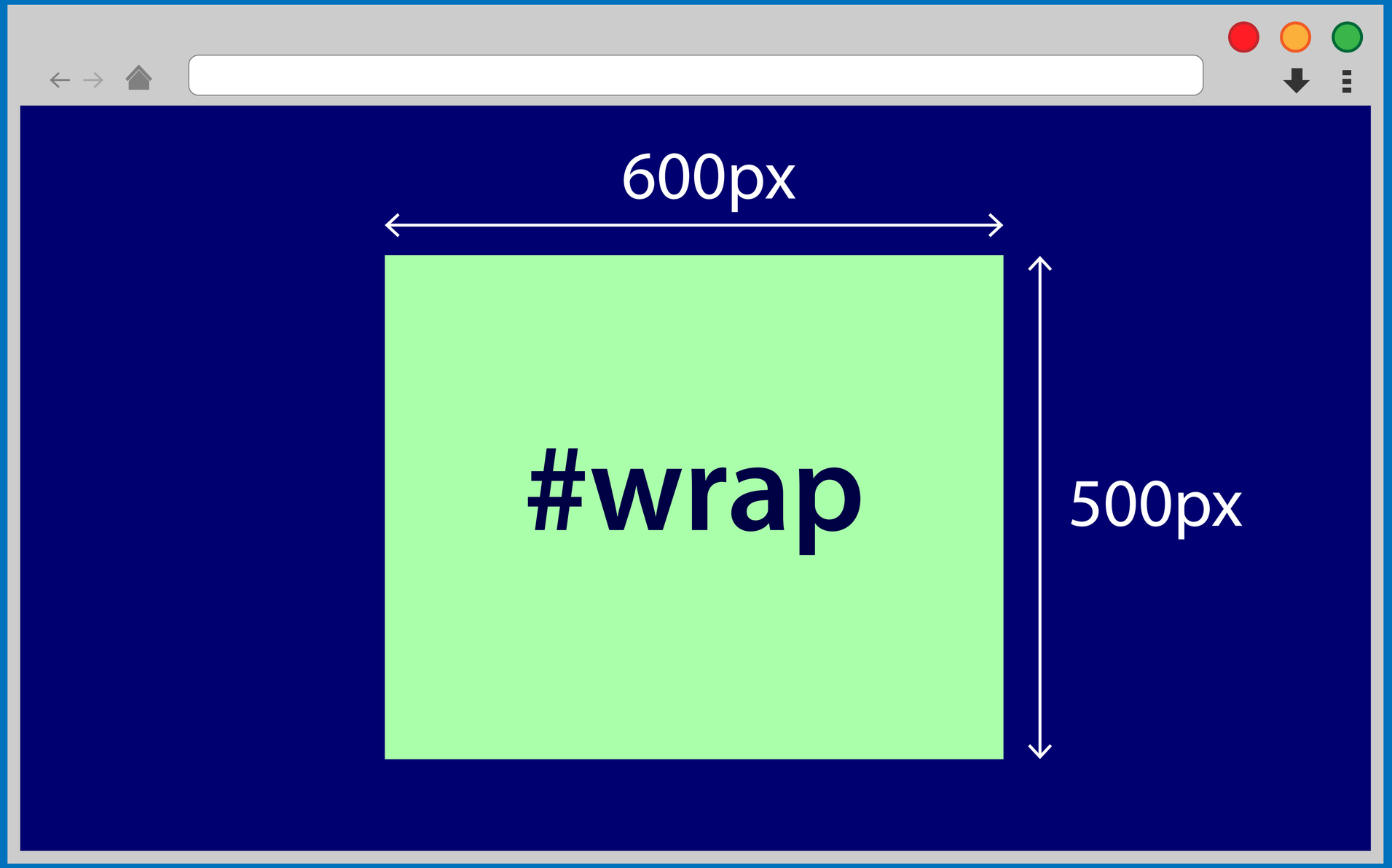
```
/* layout_01.css */


body {
  background-color: #000070;
}
```

STEP_02
layout - 01

완성 결과물 보고 제작하기

#000070

#AAFFAA



STEP_02
layout_01.html

<body> 내용에 작성

```
<body>
  <div id="wrap">
    #wrap
  </div>
</body>
```

차후 필요없는 내용일 경우
반드시 내용을 지워야
원활한 레이아웃이 제작됨

STEP_02
layout_01.css

기존 내용
뒤 작성

```
/* layout_01.css */

body {
   background-color: #000070;
}
#wrap {
   width:600px;
   height:500px;
   background-color:#aaffaa;
   margin:auto;
   margin-top:100px;
}
```

margin
여백을 의미하며,
해당요소의 공간을 만드는 역할

margin:auto;
여백을 자동으로 처리하라는 의미이며,
좌/우의 여백을 균등하게 주어
웹페이지 전체의 가운데 배치

margin-top:100px;
margin:auto; 이후에 작성해야 적용되며,
해당요소의 상단에 공백을 100px 처리

STEP_03
layout - 02

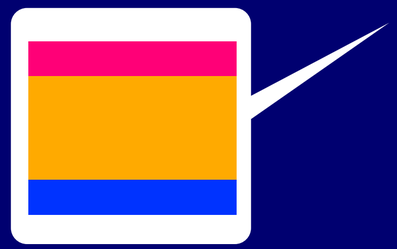
#wrap 내부 만들기

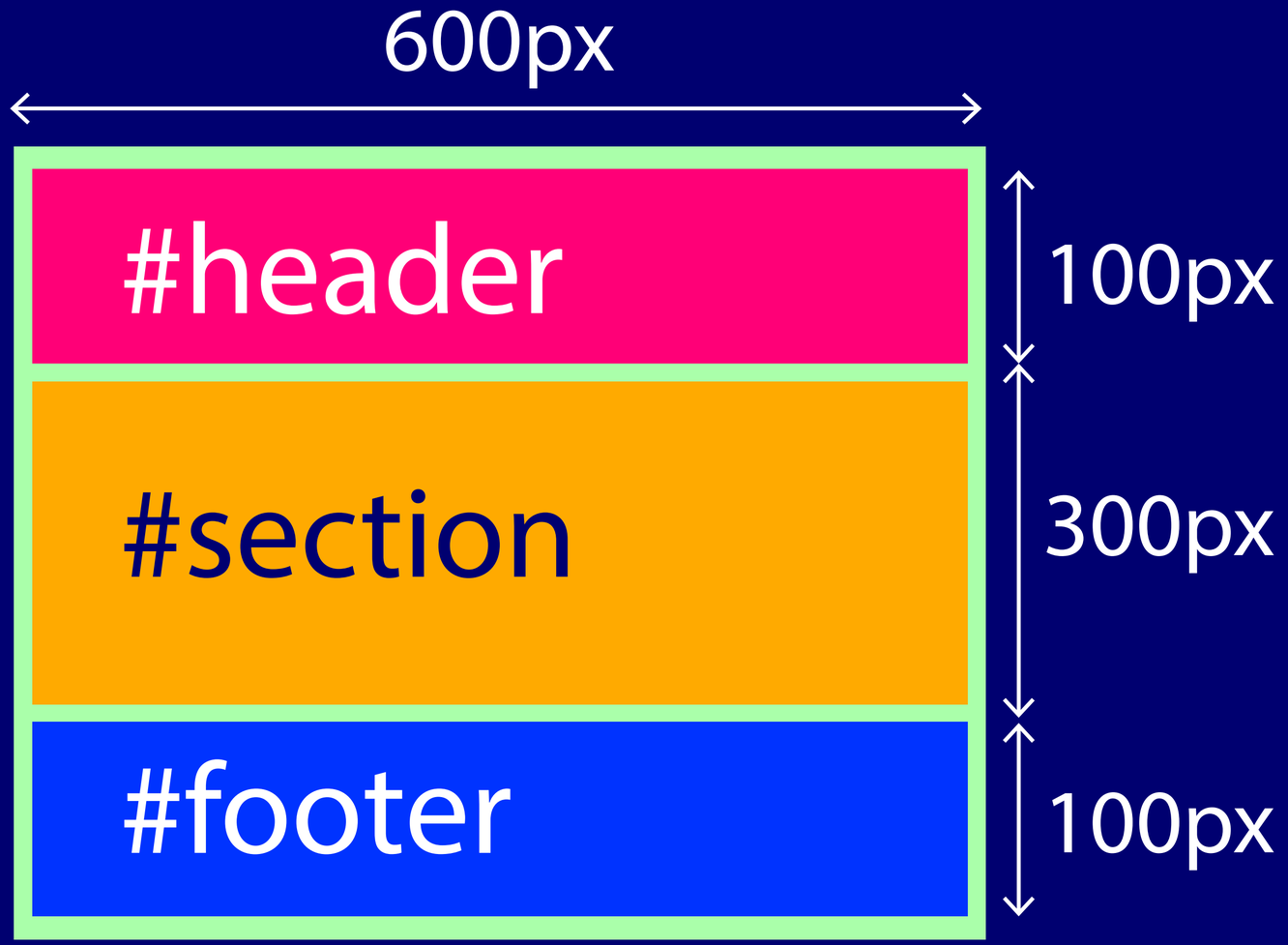
#wrap 내부
여백은 없으며,
형태를 구분하기위해
표현하였음

#FF0077

#FFAA00

#0033FF





STEP_03
layout_01.html

#wrap
내부 작성

```
<body>
  <div id="wrap">
    <div id="header">header</div>
    <div id="section">section</div>
    <div id="footer">footer</div>
  </div>
</body>
```

차후 필요없는 내용일 경우
반드시 내용을 지워야
원활한 레이아웃이 제작됨

STEP_03
layout_01.css

#wrap{} 뒤 작성

display: inline;
display: block;
display: inline-block;
display: none;

각각의 의미 파악하기

```
#wrap {...}
#header{
   width:600px; height:100px;
   background-color:#ff0077;
}
#section{
   width:inherit; height:300px;
   background-color:#ffaa00;
}
#footer{
   width:100%; height:100px;
   background-color:#0033ff;
}
```

내용 작성 후,
가려진 배경의 색상, 이미지, 글씨 등
불필요한 요소 및 내용은 삭제처리함

```
#wrap {
   width:600px; height:500px;
   margin:auto; margin-top:100px;
}
```

단위 : px, %, pt, em, rem 등 존재, 기본단위는 px이며, 정수로 처리한다.
- % : 특정조건외 부모요소의 값을 기준으로 제작
- inherit : 부모요소가 가지고 있는 수치와 동일한 값을 가지는 표현

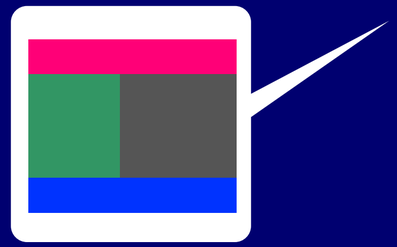
STEP_04
layout - 03

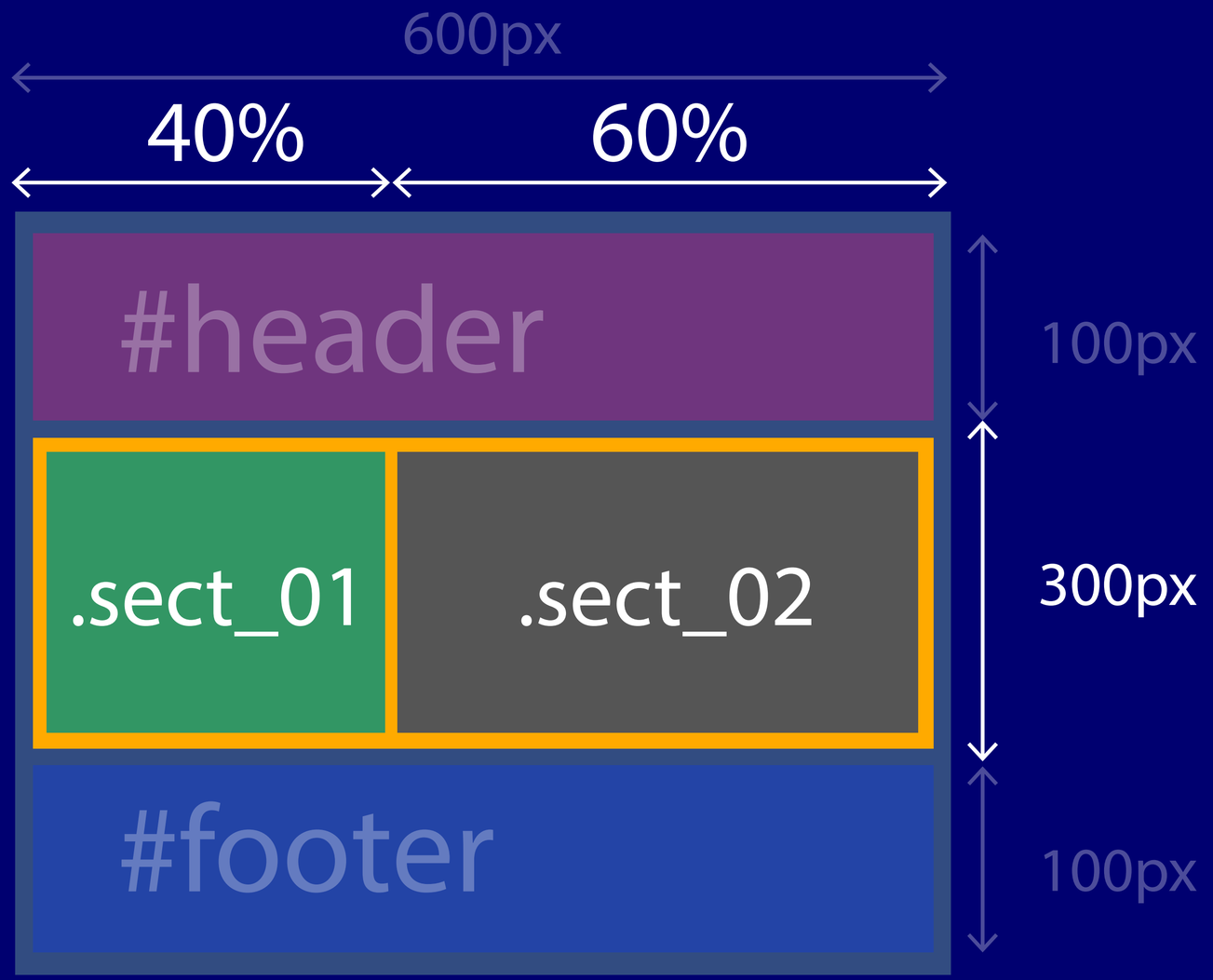
#section 내부 만들기

#section내부에
두개의 요소를 제작
두가지 결과를 제작 체크
1. 200px, 400px 형태
2. 40%, 60% 형태

#329664

#555555





STEP_04
layout_01.html

#section
내부 작성

```
<body>
  <div id="wrap">
    <div id="header">header</div>
    <div id="section">
        <div class="sect_01">sect_01</div>
        <div class="sect_02">sect_01</div>
    </div>
    <div id="footer">footer</div>
  </div>
</body>
```

차후 필요없는 내용일 경우
반드시 내용을 지워야
원활한 레이아웃이 제작됨

STEP_04
layout_01.css

#section{}
내용 뒤 작성

float:left;
float:right;
clear:both;

각각의 의미 파악하기

```
#section{...}
.sect_01 {
  float:left;
  width:40%; height:300px;
  background-color:#329664;
}
.sect_02 {
  float:right;
  width:60%; height:300px;
  background-color:#555555;
}

#footer{...}
```

float 기능을 사용하면,
이웃하는형제 요소는 모두
float기능을 사용

단, 이미지를 중심으로 글씨를
감싸는 형태를 처리하는것 제외.

STEP_05
layout - 04

#section 뒤 추가 만들기

#section요소 이후,
#aside 추가 요소 제작 및
내부 기능을 구분/분석 가능

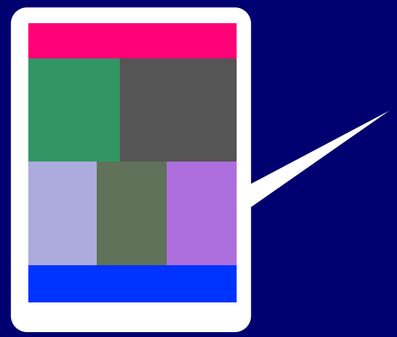
#wrap 전체 높이값 변경

#D6E700

#ADAADD

#607259

#AD6FDD



#aside

#header

100px

.sect_01

.sect_02

1/3

1/3

1/3

600px

.side_01

.side_02

.side_03

300px

#footer

100px

STEP_05
layout_01.html

#section 뒤 추가 작성

```
<div id="header">header</div>
<div id="section">section</div>
<div id="aside">
  <div class="side_01">side_01</div>
  <div class="side_02">side_02</div>
  <div class="side_03">side_03</div>
</div>
<div id="footer">footer</div>
```

#section과 #aside는 서로
형제 관계이며,
#wrap과 부모/자식 관계이다.

STEP_05
layout_01.css

#footer{}
이전 추가 작성

display: inline;
display: block;
display: inline-block;
display: none;

각각의 의미 파악하기

```
#wrap {
  ...
  height:800px;
}
.sect_02 {...}
#aside{ width:100%; height:300px; }
#aside>div{ float:left; height:100%; }
.aside_01{width:33.333333%; background-color:#adaadd;}
.aside_02{width:33.333334%; background-color:#607259;}
.aside_03{width:33.333333%; background-color:#ad6fdd;}

#footer{...}
```

#aside 내용이 추가로 구성되어
전체 높이가 높아졌으므로,
#wrap의 높이값을 변경해야함.

% 값은 부모의 크기를 기준으로 처리되면,
전체 크기를 일정 비율로 나누어 처리할 경우 최대한 정확한 %값을 표현하기 위해
소수점 6자리 까지 계산하도록 한다.
100%를 맞출 경우, 보통 가운데 요소를 늘리거나 줄여서 전체 비율을 조정한다.
calc( ) 계산법 사용시 연산기호는 반드시 띄어쓰기로 구분해야한다.

STEP_06

layout - 05-1

#aside 뒤 추가 만들기

#aside 요소 이후,
#article 추가 요소 제작 및
내부 기능을 구분/분석 가능

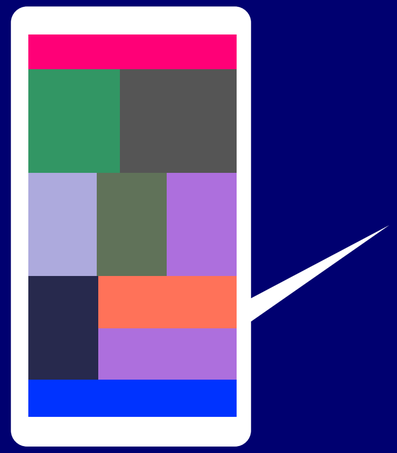
#wrap 전체 높이값 변경

#D6E700

#27294D

#FF7259

#AD6FDD



STEP_06
layout - 05-2

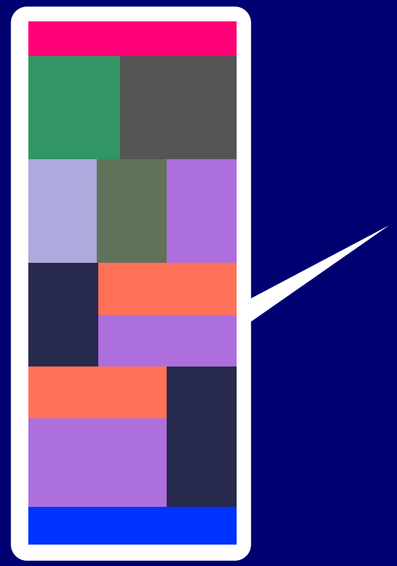
#article 내용 형태 변경하기

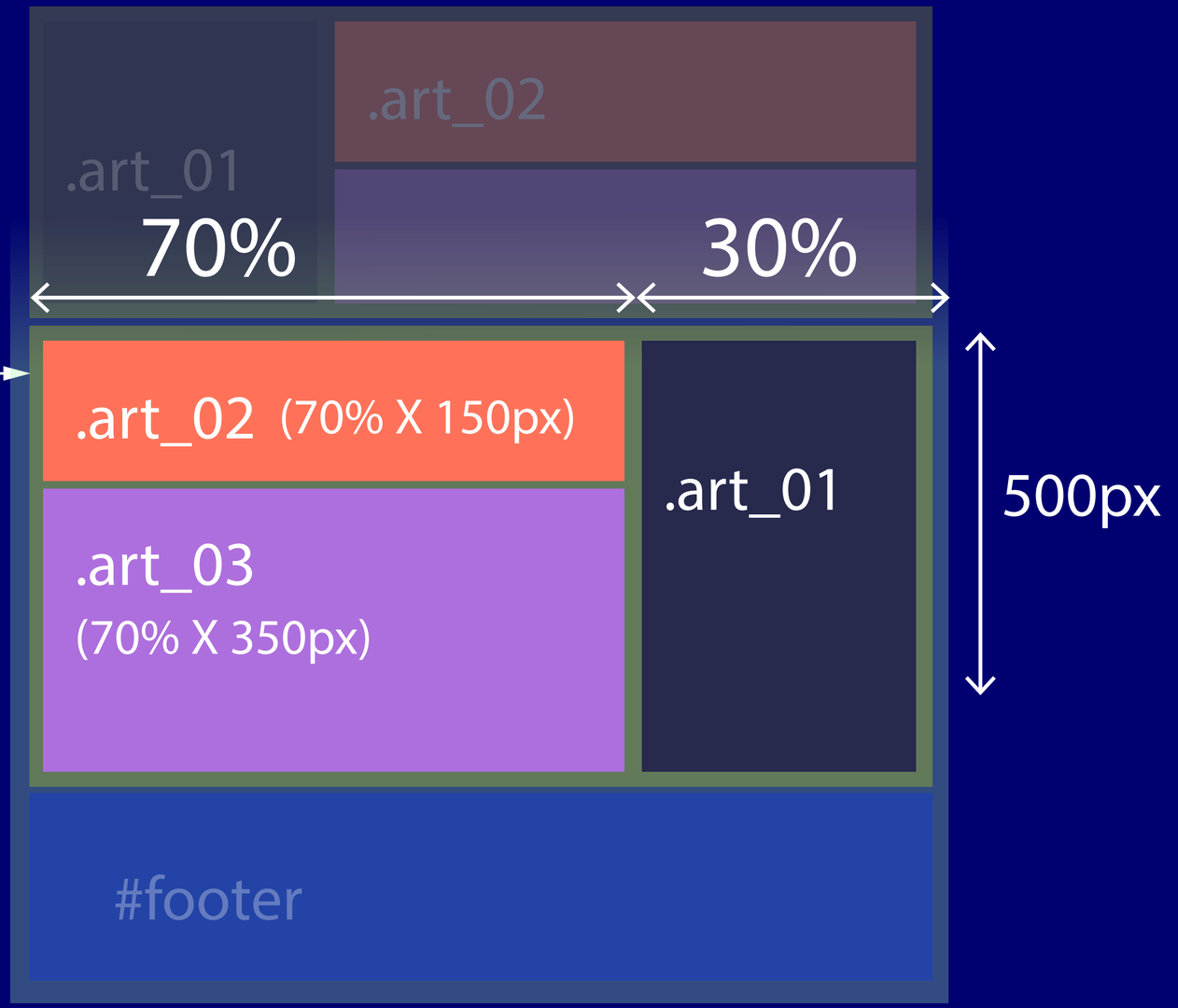
#article 추가 요소
레이아웃배치 변경

1. css float값을 변경 및
   내용읽는순서파악하기

2. html 요소 변경/
   css float값 변경

다양한 문제점 파악



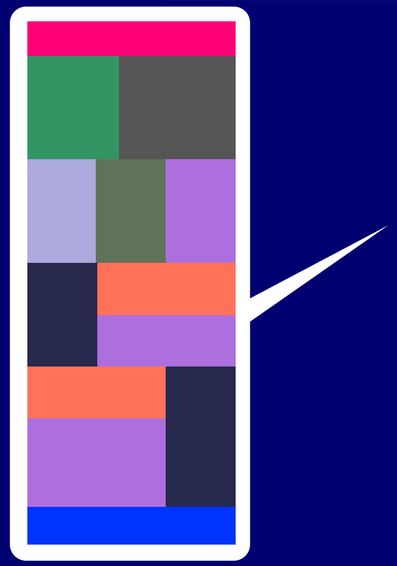


STEP_06
layout - 05-3

#article 내용 형태 변경하기

#article 추가 요소
레이아웃배치 변경

.art_02,
.art_03 요소의
float값을 지우고,
부모요소인
.art_area요소를 생성하여,
float값을 처리



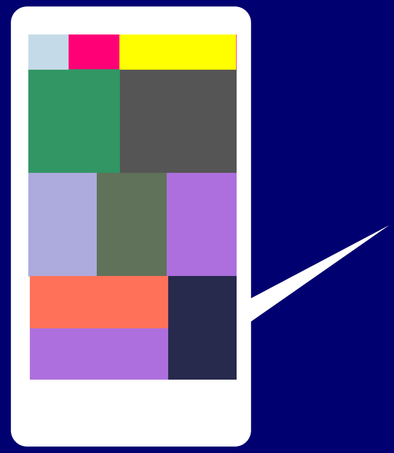
STEP_06
layout - 05-4

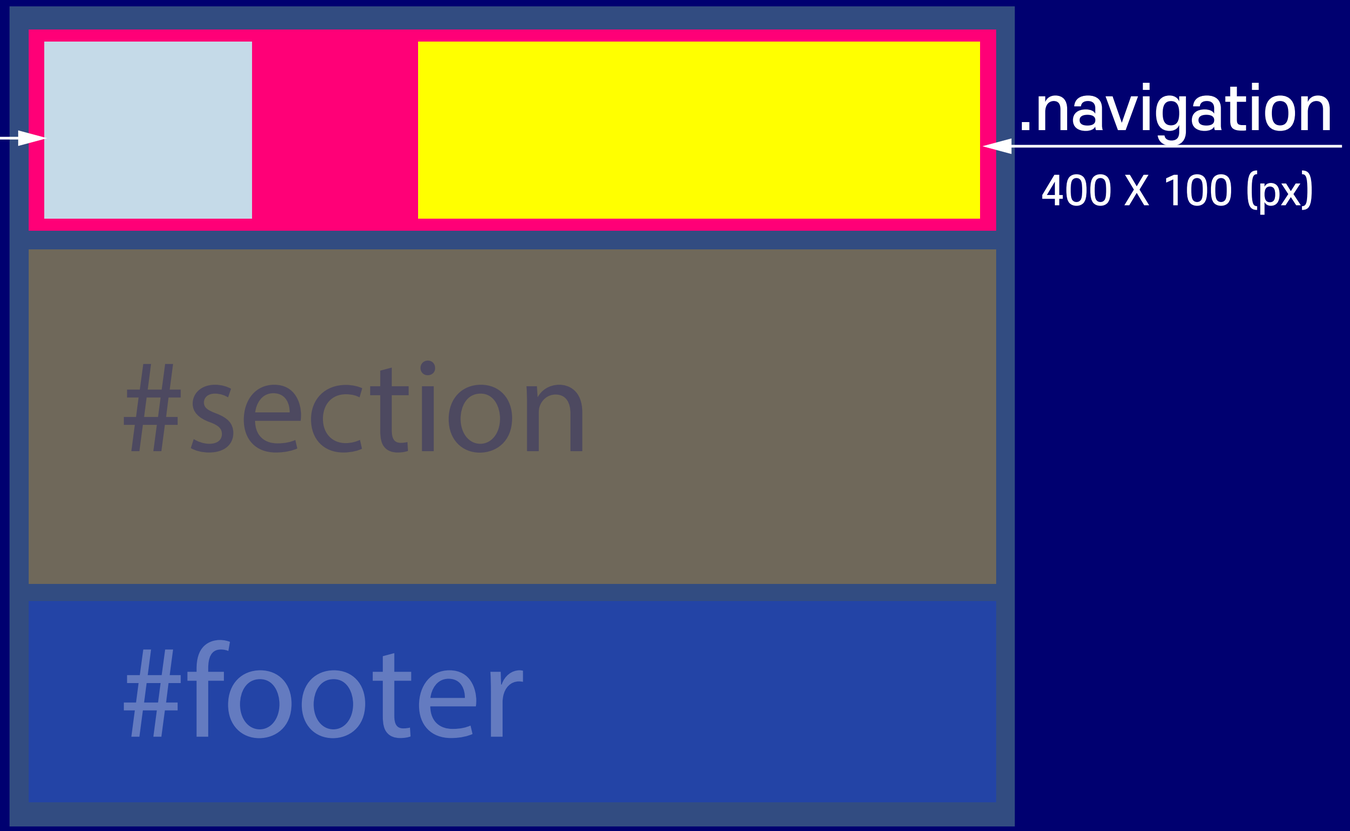
#header 내용 형태 변경하기

#header 추가 요소
레이아웃배치 변경

h1 요소는
로고/회사중요표시
.navigation 요소는
주요 메뉴 링크 영역 의미
좌우/ 위치로 인해
float값을 처리

단위표기시
가로 x 세로 순서로표기





STEP_06

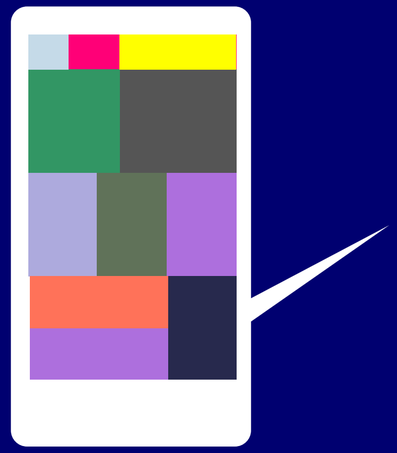
layout - 05-5

.navigation 메뉴분할하기

#header 추가 요소
레이아웃배치 변경

.navigation 요소는
여러개의 링크 항목을
가지고있으므로,
형태 및 역할을
고려해 보아야함

li 요소는 ul/ol의 요소를
부모요소로 취해야 함



STEP_06
layout - 05-6

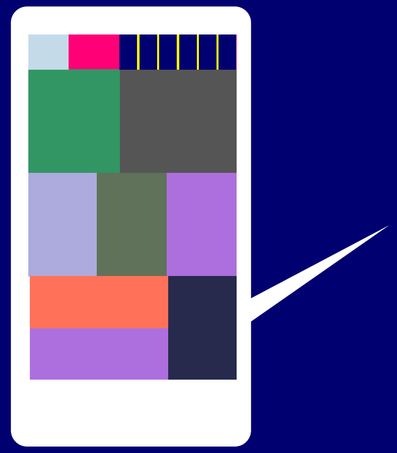
필요요소에 a / img 삽입

img, a(anker) 요소는
독단으로 사용할 수 없으며,
block 요소를
감싸고 있어야함.

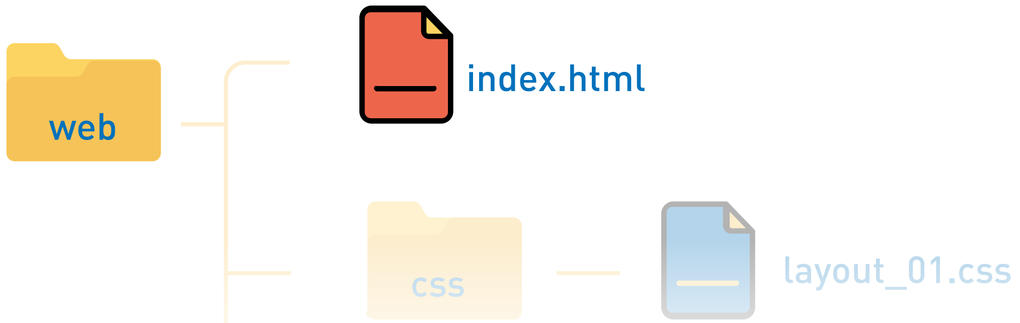
anker 요소는
inline 요소는 크기를 가질 수 없으므로, 필요시 강제로 block요소로 처리
(display : block)

padding
overflow:hidden;
overflow:visible;
overflow:auto;
overflow-x:...;
overflow-y:...;

의미 파악하기



기능 요약 및 정리



- 웹페이지 내에서 인지하는 파일은 index.html
- html/ css/ javascript 문서는 무도 각각의 기능을 처리하므로,
  별도의 파일을 생성하여 연결하는 것을 권장
- 디자인/코딩 작업시 전체의 큰 형태부터 차분히 제작해 나갈것을 권장.
- layout 위치를 파악하기 위해, float, margin 등의 각각의 기능을 파악 할것
- 여백사용시 margin/padding의 용도를 파악하여 적절히 사용할 것
- inline/block요소의 의미를 파악하고 필요시 변화하여 사용
  단, inline요소는 block요소 내부에 삽입하여 사용(block요소로 변환하여도 고유성질은 inline)
- img요소 사용시가로/세로의 크기를비유에 맞게 사용
- 넘치는 영역시 overflow 기능을 사용하여 처리

Basic layout 01 END