

CSS

HTML



Programación FrontEnd

Hojas de estilo en cascada CSS

Docente: Roman G.



Contenidos

Css

Propiedades básicas

Flex Horizontal

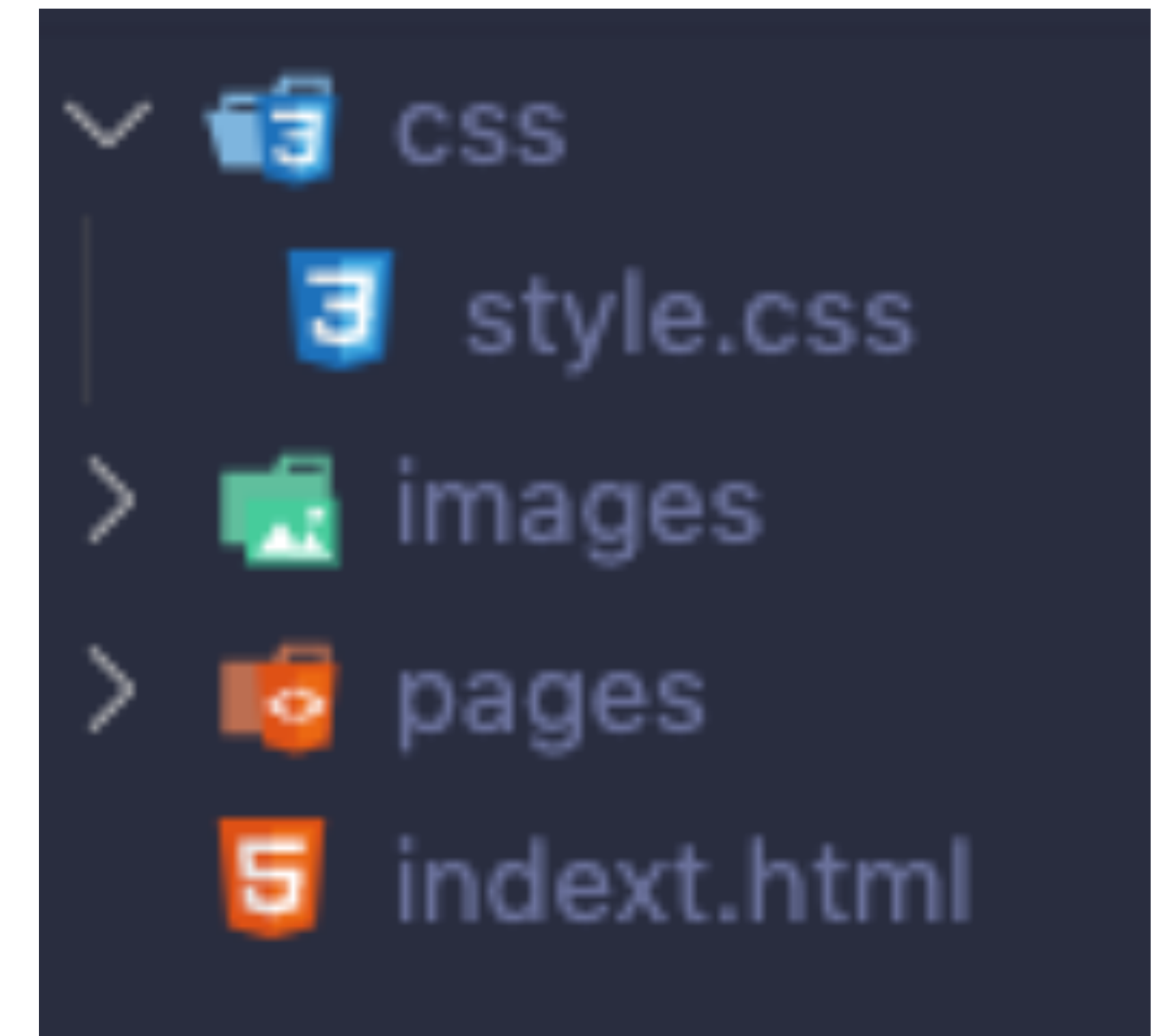
CSS

Cascading Style Sheet

Css es una hoja de estilos en cascada que sirve para dar estilo a una estructura Web

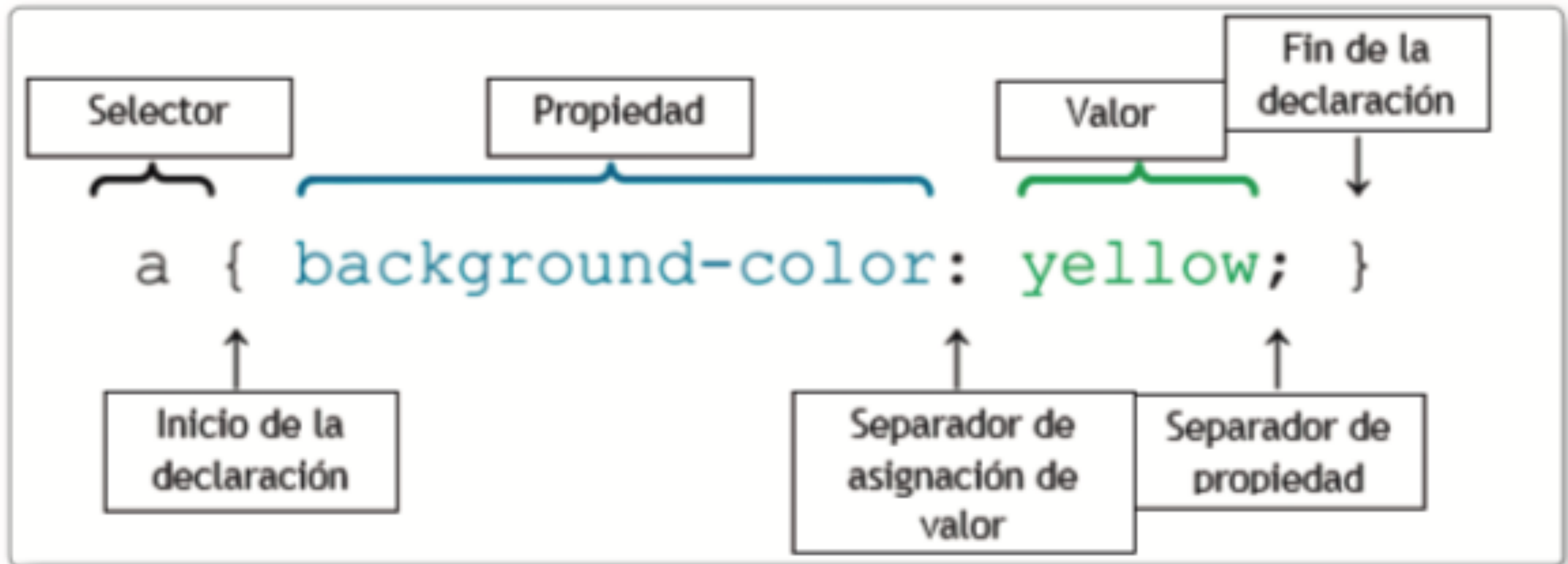
Los estilos css se aplican a las etiquetas HTML a través del style

También una etiqueta puede hacer referencia a un estilo mediante su selector, o sus atributos class o id.



Sintaxis

Cascading Style Sheet



Como se aplica Cascading Style Sheet

```
P{  
  COLOR:BLUE;  
  FONT-FAMILY:"TAHOMA";  
  FONT-SIZE: 5PT  
}
```

```
<P>CONTENIDO TEXTUAL</P>
```

Como se importa Cascading Style Sheet



```
8
9  <!doctype html>
10 <html>
11   <head>
12     <meta charset="utf-8">
13     <title>Head First Lounge</title>
14
15     <link rel="stylesheet" href="css/style.css">
16
17   </head>
18
```

Fonts

Font

Font-family

Font-size

Font-weight

```
body {  
  font-family: arial;  
  font-size: 18px;  
}
```

```
a {  
  font-weight: bold;  
}
```


Background

Background

Background-color

Background-image

Background-size

```
body {  
  font-family: arial;  
  font-size: 18px;  
  background: linear-gradient(to left, ■#ffeb3b, ■#fff59d);  
}
```

Gradiente: linear-gradient(to left, color1 , color2)

Text

Color


Text-align

Text-decoration

Text-transform

letter-spacing

text-shadow

```
a {  
  text-decoration: none;  
  font-weight: bold;  
  color:  #03a9f4;  
}  
  
h1,h2 {  
  text-shadow: 3px 3px 8px  grey;  
  text-align: justify;  
}
```

Bordes

Border

Border-left

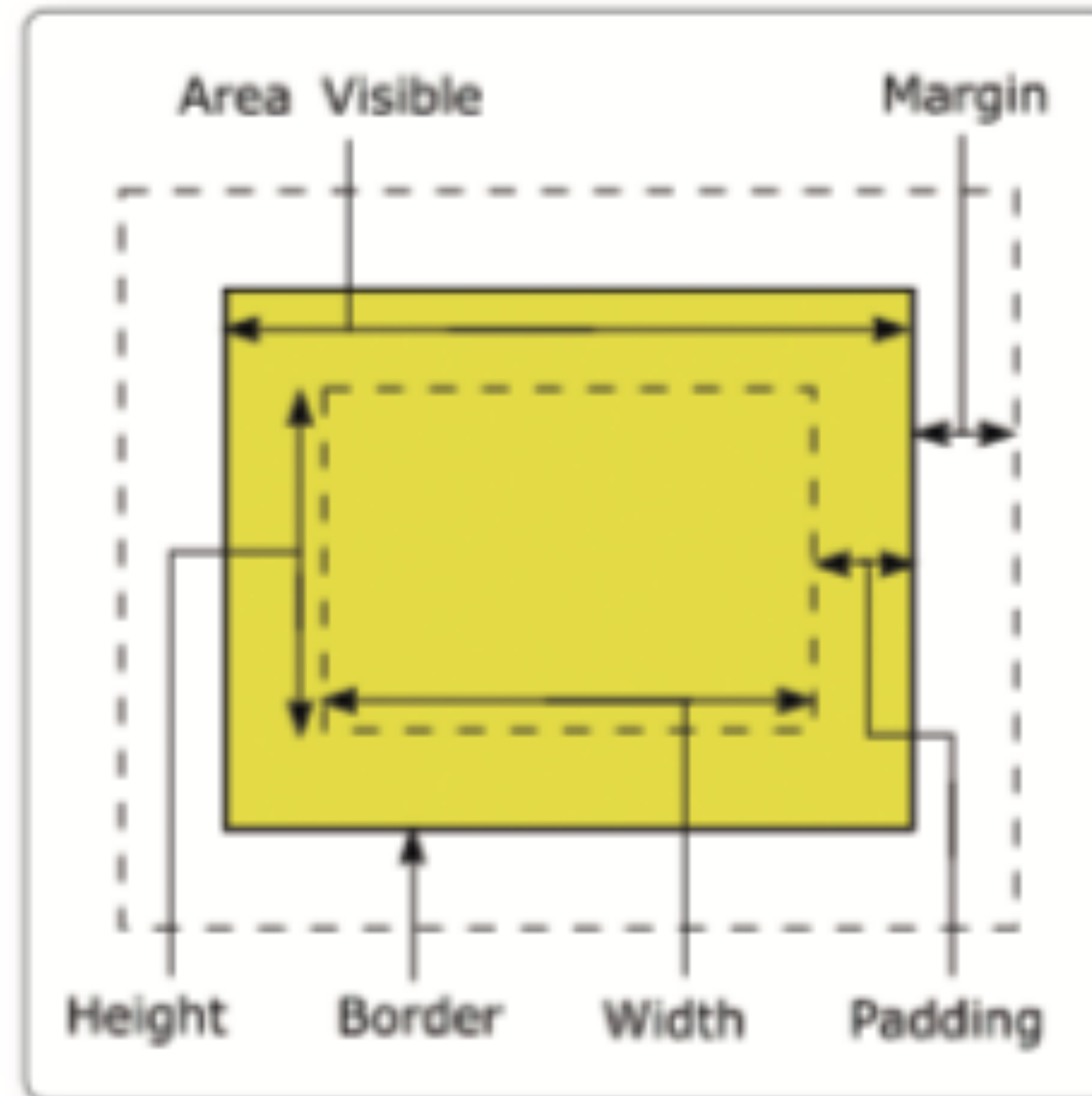
Border-right

Border-bottom

Border-top

`border: 1px solid black;`

`border-radius`



Margin

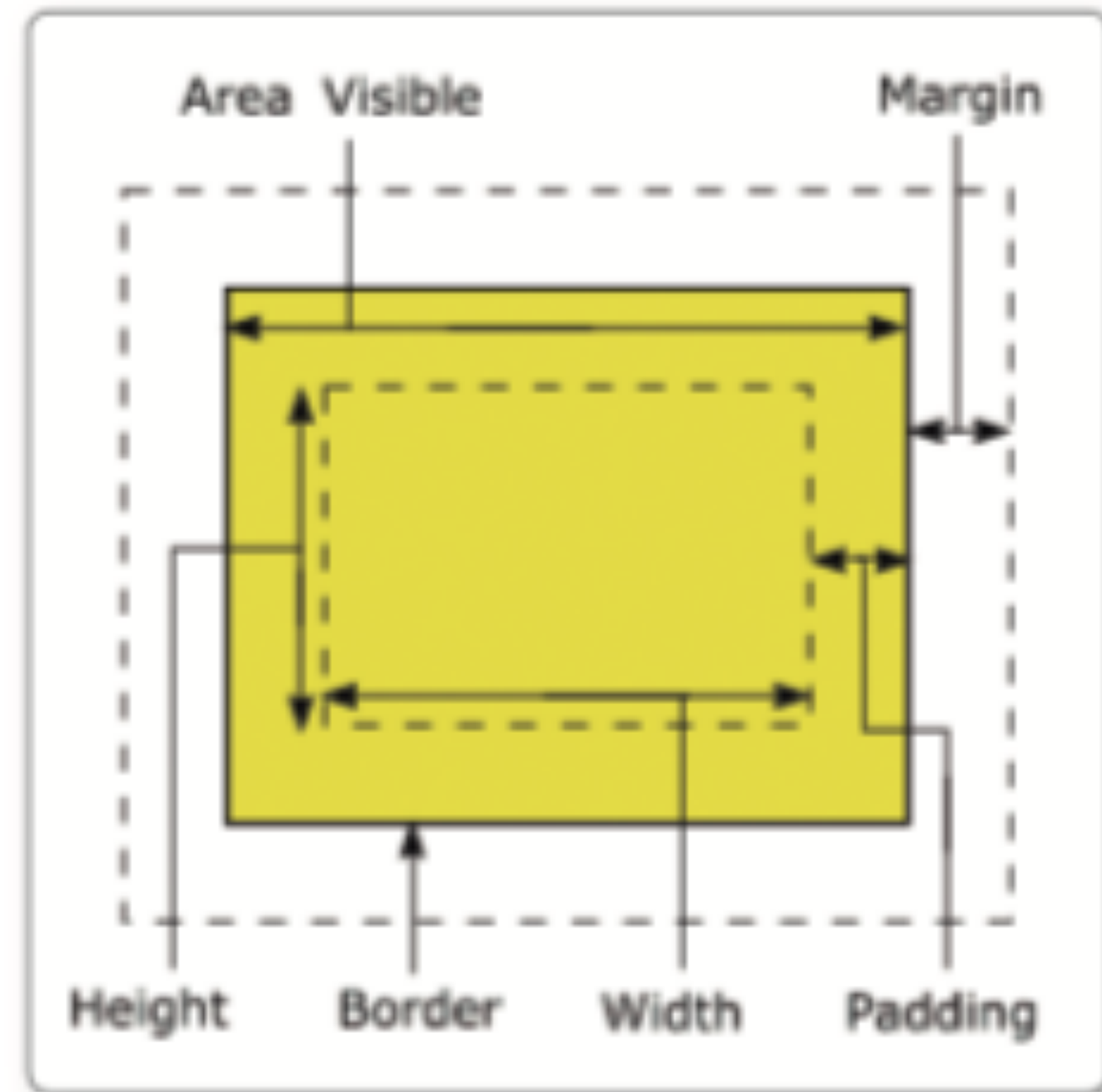
Margin

Margin-top

Margin-bottom

Margin-left

Margin-right



Padding

Padding

Padding-left

Padding-right

Padding-bottom

Padding-top

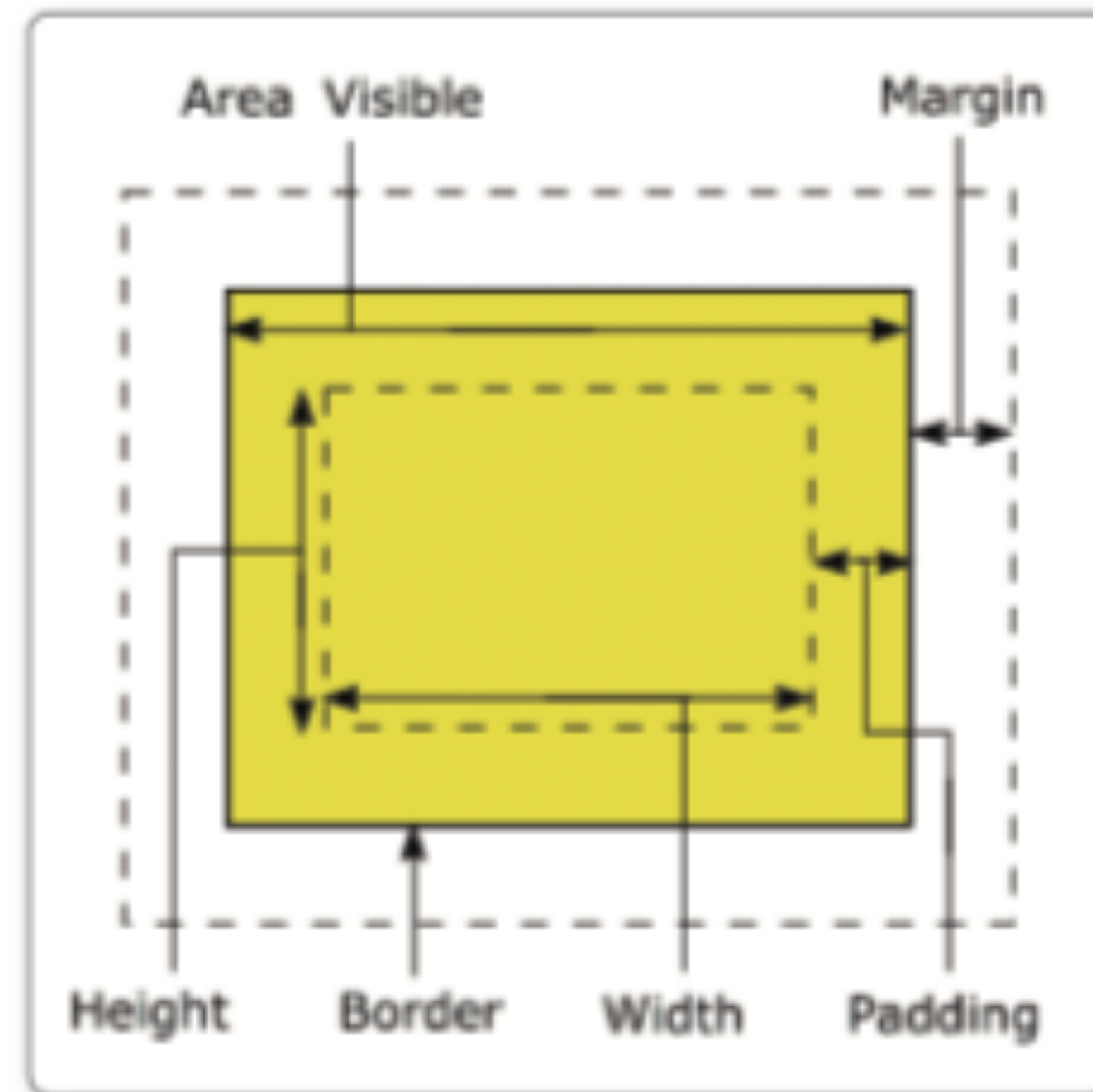


Image inside box

```
img {  
  height: 100%;  
  width: 100%;  
  object-fit: cover;  
}
```

Boxing

box-shadow

box-shadow: x y blur color

```
.caja{  
  height: 300px;  
  width: 600px;  
  box-shadow: 3px 3px 15px black;  
}
```

```
<div class="caja">  
  
</div>
```

Variables

```

:root{
  /* Primary */
  --Paleblue: hsl(225, 100%, 94%);
  --Brightblue: hsl(245, 75%, 52%);

  /* Neutral */
  --Vexpaleblue: hsl(225, 100%, 98%);
  --Desaturatedblue: hsl(224, 23%, 55%);
  --Darkblue: hsl(223, 47%, 23%);

  font-size: 16px;
  font-family: 'Red Hat Display', sans-serif;
}
p{
  background:var(--Paleblue)
}
```


Convertir etiqueta <a> en Botón

```
.boton{
  text-decoration: none;
  background-color: green;
  padding: 2px 8px;
  cursor:pointer;
  color:white;
  box-shadow: 3px 3px 15px grey;
}
```

```
<a class="boton" href="#">aceptar</a>
```

Flex

Por defecto deja los elementos de una etiqueta alineados de manera horizontal.

Posee propiedades adicionales como

flex-direction: puede ser row (horizontal) o column (vertical)

```
display: flex;  
flex-direction: row;
```

```
display: flex;  
flex-direction: column;
```

Flex

Horizontal

Cuando se trabaja con componentes de forma horizontal, puedes mover los elementos de manera horizontal con la propiedad **justify-content**

```
display: flex;  
justify-content: flex-start;
```

```
display: flex;  
justify-content: flex-end;
```

```
display: flex;  
justify-content: center;
```

```
display: flex;  
justify-content: space-around;
```

```
display: flex;  
justify-content: space-between;
```

```
display: flex;  
justify-content: space-evenly;
```

Flex

Horizontal

Cuando se trabaja con componentes de forma horizontal, puedes mover los elementos de manera vertical con la propiedad **align-items**

```
display: flex;  
align-items: flex-start;
```

```
display: flex;  
align-items: flex-end;
```

```
display: flex;  
align-items: center;
```

```
display: flex;  
align-items: stretch;
```


Maquetación

 jsstartup

#82

JavaScript Challenge

Accept or Reject - Choice is Yours

What will be the output of code below ?


```
let object = {  
  name : "JS Startup",  
  work : "JavaScript"  
};  
let result = Object.entries(object);  
  
console.log(result[0][1]); // output = 🤔 ?
```

Made with ♥ by, @jsstartup

 "JS Startup"

 "name"


 "JavaScript"

 Nested array access...


 /jsstartup


jsstartup.com

 /jsstartup



Maquetación

 jsstartup



#82


JavaScript Challenge


Accept or Reject - Choice is Yours


What will be the output of code below ?


```
let object = {  
  name : "JS Startup",  
  work : "JavaScript"  
};  
let result = Object.entries(object);  
  
console.log(result[0][1]); // output = 🤔 ?
```


Made with ♥ by, @jsstartup


 "JS Startup"


 "name"




 "JavaScript"


 Nested array access...

 /jsstartin

 jsstartin.com

 /jsstartin



Flex Vertical

Cuando se trabaja con componentes de forma vertical, puedes mover los elementos de manera vertical con la propiedad **justify-content**

```
display: flex;  
justify-content: flex-start;
```

```
display: flex;  
justify-content: flex-end;
```

```
display: flex;  
justify-content: center;
```

```
display: flex;  
justify-content: space-around;
```

```
display: flex;  
justify-content: space-between;
```

```
display: flex;  
justify-content: space-evenly;
```


Flex

Vertical

Cuando se trabaja con componentes de forma vertical, puedes mover los elementos de manera horizontal con la propiedad **align-items**

```
display: flex;  
align-items: flex-start;
```

```
display: flex;  
align-items: flex-end;
```

```
display: flex;  
align-items: center;
```

```
display: flex;  
align-items: stretch;
```

Grid

Otra forma de ordenar los componentes es por medio de grillas en donde algunas de las propiedades se destacan:

grid-template-columns





grid-template-rows

grid-gap

```
display: grid;  
grid-template-columns: 2fr 1fr;  
grid-gap: 5px;
```

Variables

Podemos definir variables en Css para reutilizar dentro del mismo diseño

```
:root {  
  --Cyan:  hsl(179, 62%, 43%);  
  --BrightYellow:  hsl(71, 73%, 54%);  
  --LightGray:  hsl(204, 43%, 93%);  
  --GrayishBlue:  hsl(218, 22%, 67%);  
  --fontSize: 16px;  
}
```

```
body {  
  font-size: var(--fontSize);  
  font-family: "Karla", sans-serif;  
  background: var(--LightGray);  
  margin: 0;  
  width: 100%;  
  height: 100%;  
}
```

Responsive

Puedes elegir el diseño para una cierta resolución, por ejemplo en el código tomará el diseño cuando la resolución sea de 768px o más.

```
@media screen and (min-width: 768px) {  
  .info {  
    display: flex;  
    justify-content: center;  
  }  
  .card-sub {  
    width: 50%;  
    border-radius: 0 0 0 8px;  
  }  
  .card-why {  
    width: 50%;  
    border-radius: 0 0 8px 0;  
  }  
}
```