

MANAGEMENT INFORMATION SYSTEMS ASSOCIATION

HTML & CSS Workshop

MISA eServices

Workshop Outline

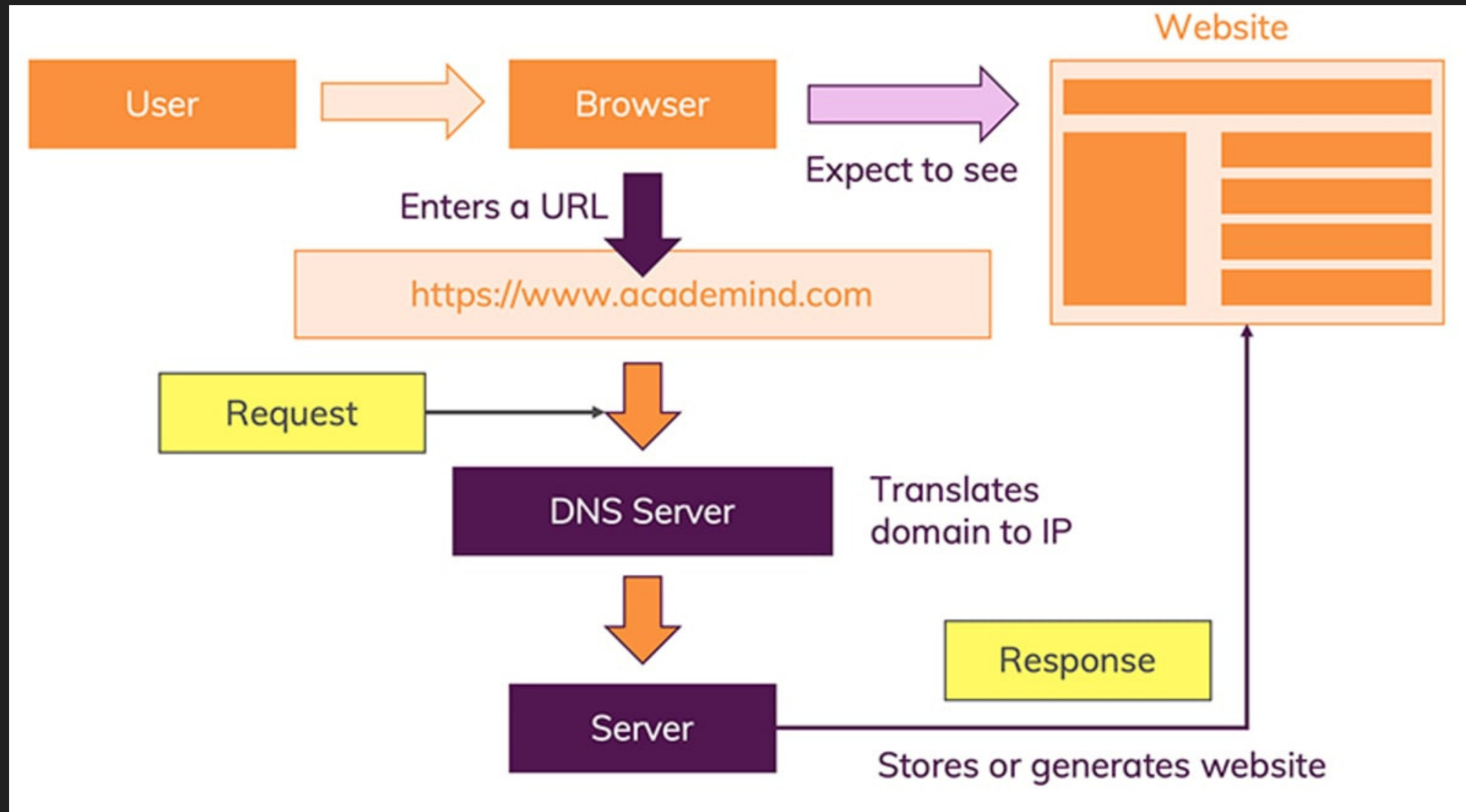
- The Web
- HTML Boilerplate & Syntax
- Common HTML Tags
 - Headings
 - Paragraphs
 - Bold / Italicize
 - Links & Images
 - Tables
 - Lists
 - Forms
 - Divs / Spans
- Semantic HTML
- CSS Syntax
- CSS Selectors (IDs & Classes)
- Specificity
- Colors & Typography
- Background Properties
- Box Model
- Positioning
- CSS Flexbox
- Media Queries
- CSS Grid
- Transitions and Animations

YOU DONT NEED TO
MEMORIZE EVERYTHING!!

Professional Developers still search
in Google for help.

Everything is one search away. Don't
be intimidated by the amount of info

How The Web Works



Source: Academind

HTML Syntax

Source: ILoveCoding.org

① Syntax to write an HTML element

Opening Tag

Every element has an opening tag with the name of the element at its start.

Attribute and its value (optional)

Attributes are like options of an element. Attributes have value.



```
<name attr="value">  
...children  
</name>
```

The diagram illustrates the syntax of an HTML element. It shows a code snippet with four arrows pointing to its components: an arrow from the 'Opening Tag' text points to the opening tag '<name'; an arrow from the 'Attribute and its value' text points to the attribute 'attr="value"'; an arrow from the 'Children (optional)' text points to the text '...children'; and an arrow from the 'Closing Tag' text points to the closing tag '</name>'.

Closing Tag

A closing tag has the name of the element with a forward slash "/" before it.

Children (optional)

Between the opening and closing tags are the children of the element. This can be more elements or just plain text.

Common HTML Tags

Heading Tags

- `<h1>Heading 1</h1>`
- `<h2>Heading 2</h2>`
-
- `<h6>Heading 6</h6>`

Paragraph Tags

- `<p>Paragraph Tag</p>`
- **Note:** Extra spaces/Line breaks do not take effect

Bold/Italicize

- `Bold`
- `Italicize`

Common HTML Tags

Links

- Can be used for linking a website url, an element with an id, or another html file
- `link text`
- `Google`

Images

- Can be used for inserting an image via local files or image url
- ``

Common HTML Tags

Tables

- `<table>` - creates table
- `<tr>` - create a table row
 - `<th>` - create table headers
 - `<td>` - create table data

```
<table>
  <tr>
    <th>Firstname</th>
    <th>Lastname</th>
    <th>Age</th>
  </tr>
  <tr>
    <td>Jill</td>
    <td>Smith</td>
    <td>50</td>
  </tr>
</table>
```

Common HTML Tags

Unordered List

```
<ul>  
  <li>Coffee</li>  
  <li>Tea</li>  
  <li>Milk</li>  
</ul>
```

Ordered List

```
<ol>  
  <li>Coffee</li>  
  <li>Tea</li>  
  <li>Milk</li>  
</ol>
```

Common HTML Tags

Forms & Inputs

```
<form action="/login" method="POST">  
  <label for="fname">Name:</label>  
  <input type="text" id="fname" name="fname">  
  
  <label for="pass">Password</label>  
  <input type="password" id="pass" name="pass">  
  
  <input type="submit" value="Submit">  
</form>
```

Forms & Inputs

Common Input Types

- `<input type="text">`
- `<input type="email">`
- `<input type="password">`
- `<input type="color">`
- `<input type="number">`
- `<input type="checkbox">`
- `<input type="radio">`

Select Tag

```
<select name="cars" id="cars">  
  <option value="volvo">Volvo</option>  
  <option value="saab">Saab</option>  
  <option value="mercedes">Mercedes</option>  
  <option value="audi">Audi</option>  
</select>
```

Forms & Inputs

Text Area

```
<label for="desc">Description:</label>
```

```
<textarea id="desc" name="desc" rows="4" cols="50">
```

```
  Text Area text here
```

```
</textarea>
```

Block vs Inline

- A block-level element always starts on a new line. It takes up 100% of screen width.
 - i.e. Headings, Paragraph, Table, Lists, Divs
- An inline element does not start on a new line. It only takes up the width of its current content size.
 - i.e. Images, Links, Inputs, Bold/Italicize, TextArea

Common HTML Tags

Divs

- Block-level element often used as a container
- Used for dividing the document to separate parts
- Easily style these divided parts at once (i.e. every article is styled)

```
<div>  
    <h1>Article 1</h1>  
    <p>Content 1</p>  
</div>
```

```
<div>  
    <h1>Article 2</h1>  
    <p>Content 2</p>  
</div>
```

Common HTML Tags

Spans

- The `` element is an inline container used to mark up a part of a text, or a part of a document
- Similar concept with `divs` except it is inline, such as separating text styling.

```
<p>My mother has  
<span style="color:blue;">blue</span>  
eyes and my father has  
<span style="color:green;">dark  
green</span> eyes.  
</p>
```

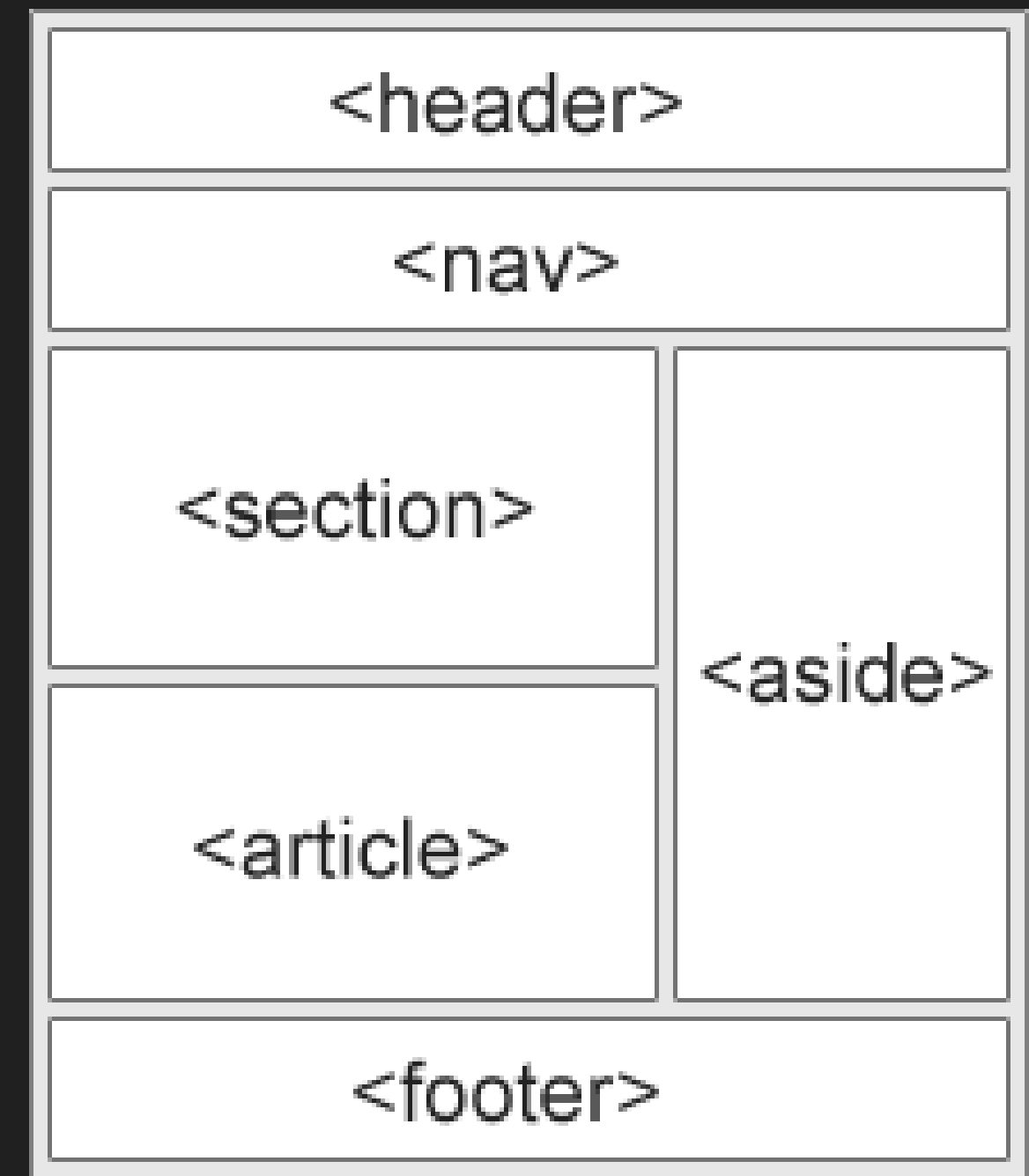

Semantic HTML

- Using `<div>` too much can lead to less readability in code
 - Which part is the navbar? where is the header? I DONT UNDERSTAND!!!
- Solution: HTML5 brought Semantic HTML
 - Exactly just like divs except semantic tags are now labeled with layout names.
 - `<header></header>` | `<footer></footer>` |
`<section></section>` | `<main></main>` |
`<article></article>` | `<nav></nav>`

Semantic HTML

Source: w3schools

Tag	Description
<u><article></u>	Defines independent, self-contained content
<u><aside></u>	Defines content aside from the page content
<u><details></u>	Defines additional details that the user can view or hide
<u><figcaption></u>	Defines a caption for a <figure> element
<u><figure></u>	Specifies self-contained content, like illustrations, diagrams, photos, code listings, etc.
<u><footer></u>	Defines a footer for a document or section
<u><header></u>	Specifies a header for a document or section
<u><main></u>	Specifies the main content of a document
<u><mark></u>	Defines marked/highlighted text
<u><nav></u>	Defines navigation links
<u><section></u>	Defines a section in a document
<u><summary></u>	Defines a visible heading for a <details> element
<u><time></u>	Defines a date/time



Semantic HTML

```
<div>
  <div>
    <h1>Article 1</h1>
  </div>
  <div>
    <p>Content 1</p>
  </div>
</div>
```

```
<div>
  <div>
    <h1>Article 2</h1>
  </div>
  <div>
    <p>Content 2</p>
  </div>
</div>
```

VS

```
<article>
  <header>
    <h1>Article 1</h1>
  </header>
  <div>
    <p>Content 1</p>
  </div>
</article>
```

```
<article>
  <header>
    <h1>Article 2</h1>
  </header>
  <div>
    <p>Content 2</p>
  </div>
</article>
```

CSS Syntax

Source: ILoveCoding.org

① Syntax to write CSS

Selectors

The element(s) on which the style should be applied

Property and its value

This is the actual style to be applied to the element(s)



```
selectors {  
    ....property: value;  
}
```

The diagram illustrates the CSS syntax with two hand-drawn arrows. One arrow originates from the 'Selectors' definition and points to the word 'selectors' in the code block. The second arrow originates from the 'Property and its value' definition and points to the 'property: value;' line in the code block. The code block is highlighted in a light gray background.

Places to write CSS

- Inline CSS - Inside the element.
Worst practice and will create redundancy.
- Internal Styling - Outside element but inside HTML file.
Not the best practice. Only good when CSS is short.
- External Styling - Best Practice.
Separates your CSS file from HTML file. Less verbose code.

Source: ILoveCoding.org

② 3 places to write CSS

(A) Inline styles

```
<element style="property: value;">
```

(B) In the <style> element

```
<head>
... <style>
... selectors { property: value; }
... </style>
</head>
```

(C) In a dedicated file (style.css) & refer that file via the <link> element

```
<head>
... <link rel="stylesheet"
... href="style.css" />
</head>
```

CSS IDs & Classes

- class - used for styling many elements at once
 - Uses a dot as a selector

HTML —

```
<p class="content">Blue Content 1</p>
```

```
<p class="content">Blue Content 2</p>
```

CSS — .content {color: blue;}

- id - used for styling one unique element
 - Uses a hash as a selector

HTML — <h1 id="heading">Red Header</h1>

CSS — #heading { color: red; }

Tip: Don't use IDs too much. Its best to use classes whenever possible so we can reuse our code more. Only use IDs if the element styling is surely surely surely unique. :)

CSS Selectors

Source: ILoveCoding.org

③ Selectors and their syntax

Basic Selectors

elementname

.classname

#idname

[attr=value]

*

Combinators

selectorA + selectorB Adjacent sibling

selectorA ~ selectorB General sibling

parent > child Direct child

parent descendent Descendent

Pseudo Selectors

:active

:hover

:visited

:focus

CSS Colors and Fonts

Colors

- color - changes color of children text
 - `h1 { color: blue; }`
- background-color - changes color of background
 - `div { background-color: red; }`
- Mainly uses built-in colors, hexadecimals, or rgb values

Typography

- font-family - changes font of text
 - `h1 { font-family: Arial, Helvetica, sans-serif; }`
 - You can add multiple fonts as back-ups.
- font-size - changes size of text
- font-weight - changes weight/boldness of text
- text-align - aligns text

CSS Units

Static

- px - 1/96 of an inch

Dynamic

- % - percentage relative
 - if fontsize, it is percentage relative to parent font size
 - if margin, padding, it is percentage relative to width of element
- rem - relative to root font size
 - 1rem = root font size, 0.5rem = half of root font size
- em - relative to parent font size
 - 1em = parent font size

Specificity

- Some CSS selectors are more specific than others
- Suppose two CSS selectors are assigned to one element, the more specific selector will overwrite the properties of the less specific selector.
- `<p class="par">Paragraph</p>`
 - `p {color: red; font-size: 18px}` VS `.par { color: blue; }`
- If there is no conflict in some properties of the least specific selector vs the more specific selector, the styles of former will still apply. (i.e. font-size)

Order from least to most specific

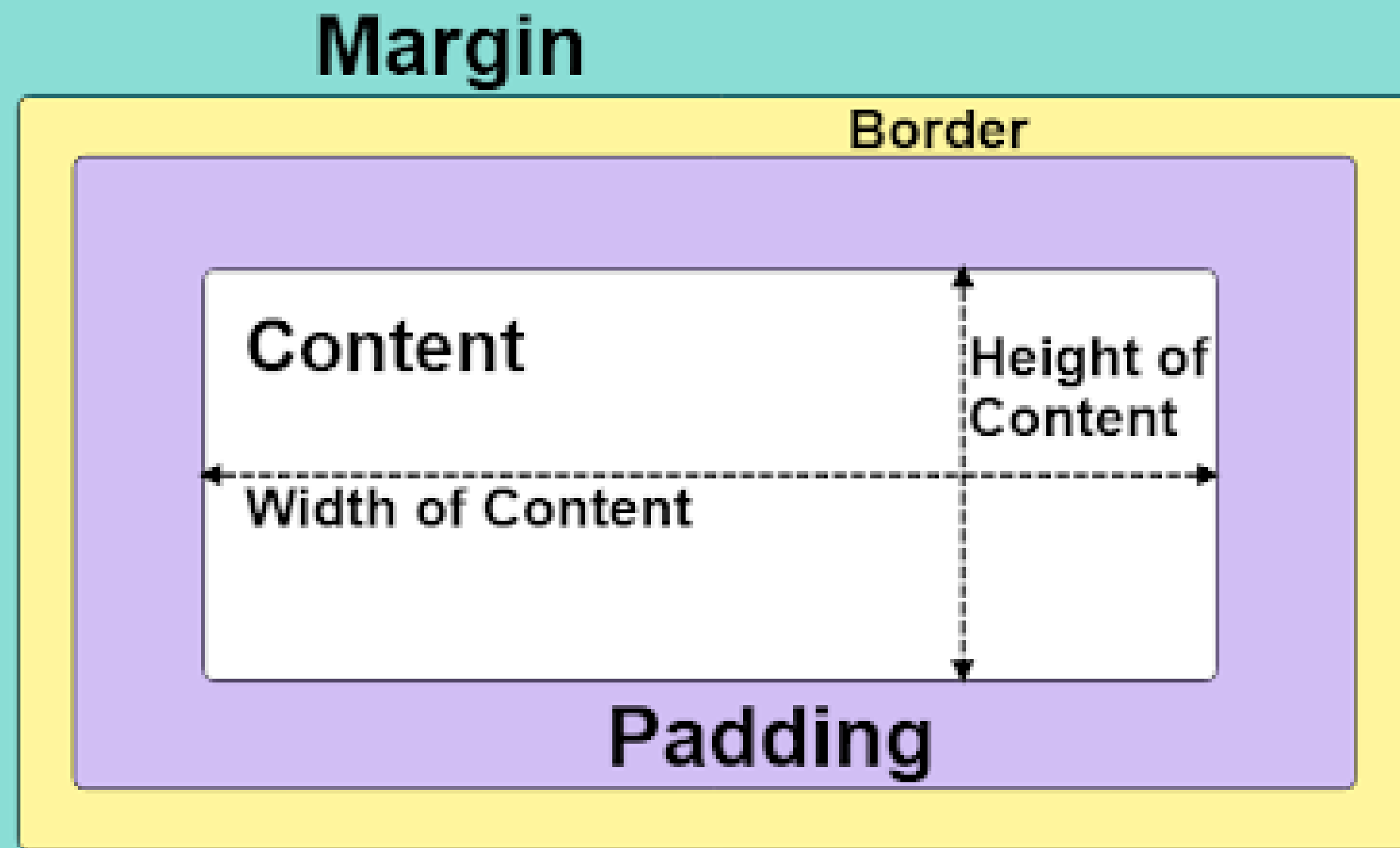
Element Tag Selectors > Classes > IDs > Inline Styles

Background Styling

- background-image - adds a background image
 - `div { background-image: url("filename.jpg"); }`
- background-repeat - whether the image will repeat in the bg
 - `div { background-repeat: no-repeat; }`
- background-position - where the origin point of the image lies
 - `div { background-position: center center; }`
 - `div { background-position: right top; }`
- background-size - specifies the size of images
 - `div { background-size: cover; }`
 - `div { background-size: contain; }`
- background - background property shorthand
 - `div { background: url("filename.jpg") center center no-repeat center center/cover; }`

Box Model

Box Model
W. Jollymore
Fall 2019
Syst10049

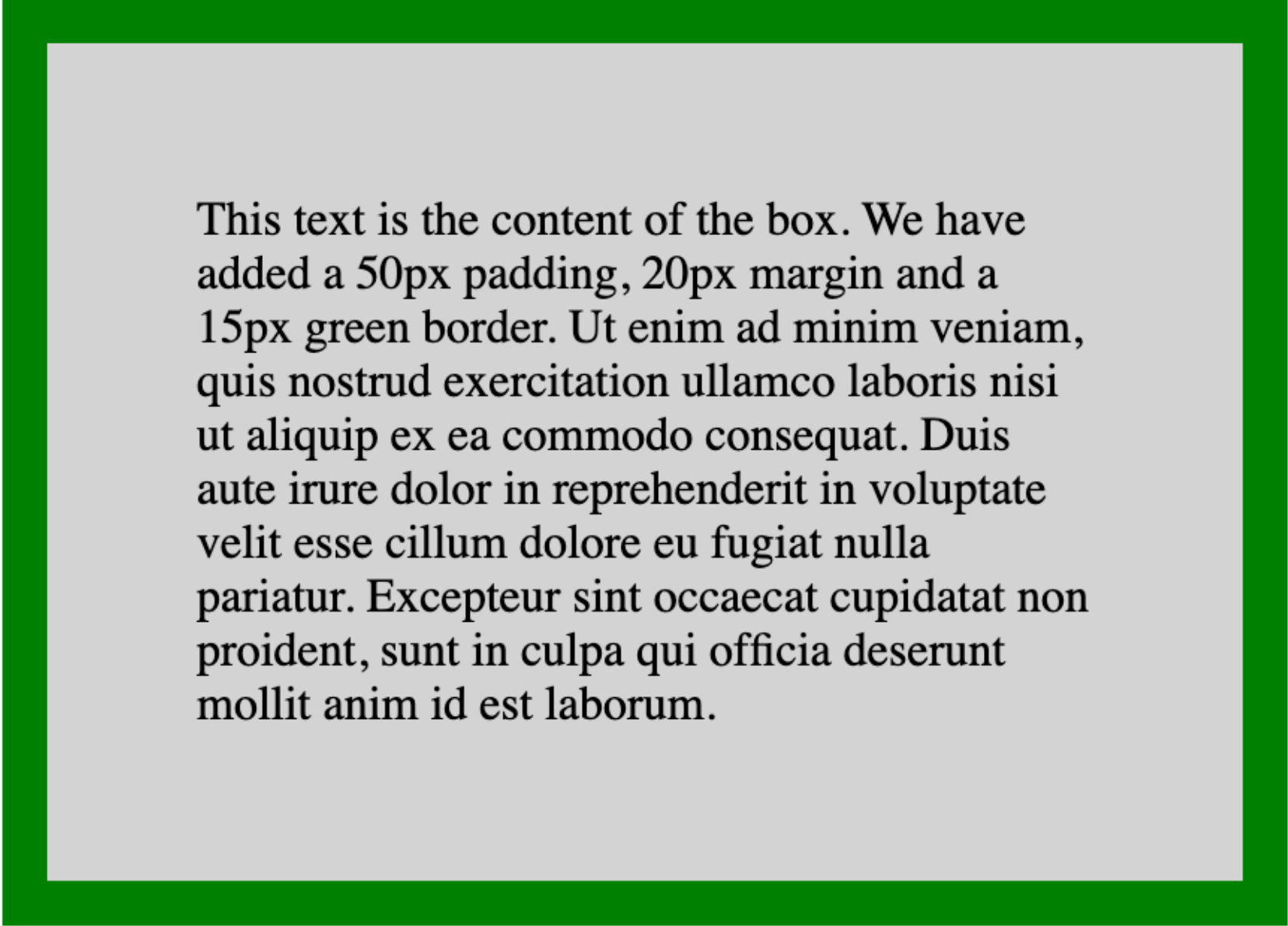


Box Model

- height / width - height/width of content
 - default is set to auto. only sizes up based on the content it holds
 - if you use %, it is relative to the height/width of parent element
 - if a parent element's height is 300px and u set a child's height to 100%, the child's height will be 300px
- padding - distance from content to border
- border - the edge of the element
- margin - distance between the current element and neighboring elements

Box Model

```
div {  
  width: 300px;  
  border: 15px solid green;  
  padding: 50px;  
  margin: 20px;  
}
```



This text is the content of the box. We have added a 50px padding, 20px margin and a 15px green border. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

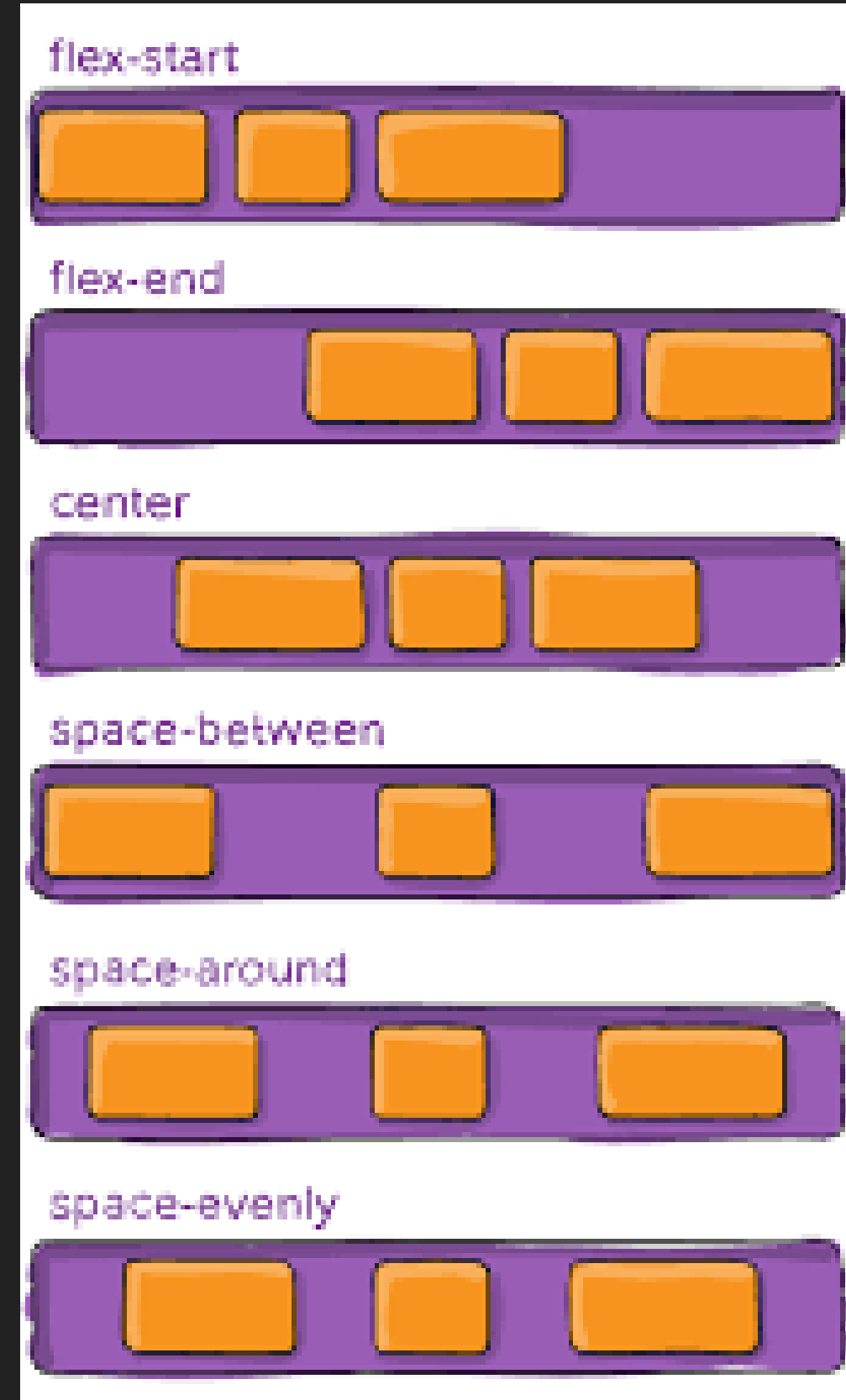
Source: w3schools.

Position Properties

- Use top, bottom, left, right properties to adjust positioned elements
- position: static - Set by default. positioned according to the normal flow of the page.
- position: relative - positioned relative to its normal position.
- position: fixed - stays in the same place even if the page is scrolled.
- position: absolute - positioned relative to the nearest positioned ancestor. However; if an absolute positioned element has no positioned ancestors, it uses the document body
- z-index - adjusts the stack order of elements.
 - Higher z-index means it has more priority in going front

Flexbox

- The Flexible Box Layout Module, makes it easier to design flexible responsive layout structure without using float or positioning.



Source: CSS Tricks

Flexbox

- If a parent is set to display: flex, its direct children becomes movable through the parent's flex properties.
- justify-content
 - this is used to move the elements in the main axis
- align-items
 - this is used to move the elements in the cross axis
- flex-direction
 - determines the main axis (column/row)
- flex-wrap
 - if not enough space, move to next line

```
<div class="flex-container">  
  <div>1</div>  
  <div>2</div>  
  <div>3</div>  
</div>
```

Media Queries

- Primarily used for responsive design
 - Being able to use the website through tablets and mobile phones
- Syntax
 - `@media (max-width: 500px) { }`
 - max-width - less than or equal to
 - min-width - greater than or equal to
 - specify changes inside brackets

CSS Grid

- Parent properties
 - grid-template-columns
 - how many columns? what's their size?
 - grid-template-rows
 - how many rows? what's their size?
 - grid-auto-rows
 - default size of a row
 - gap
 - space between each column/row
- Child properties
 - grid-column
 - what columns will I occupy?
 - grid-row
 - what rows will i occupy?

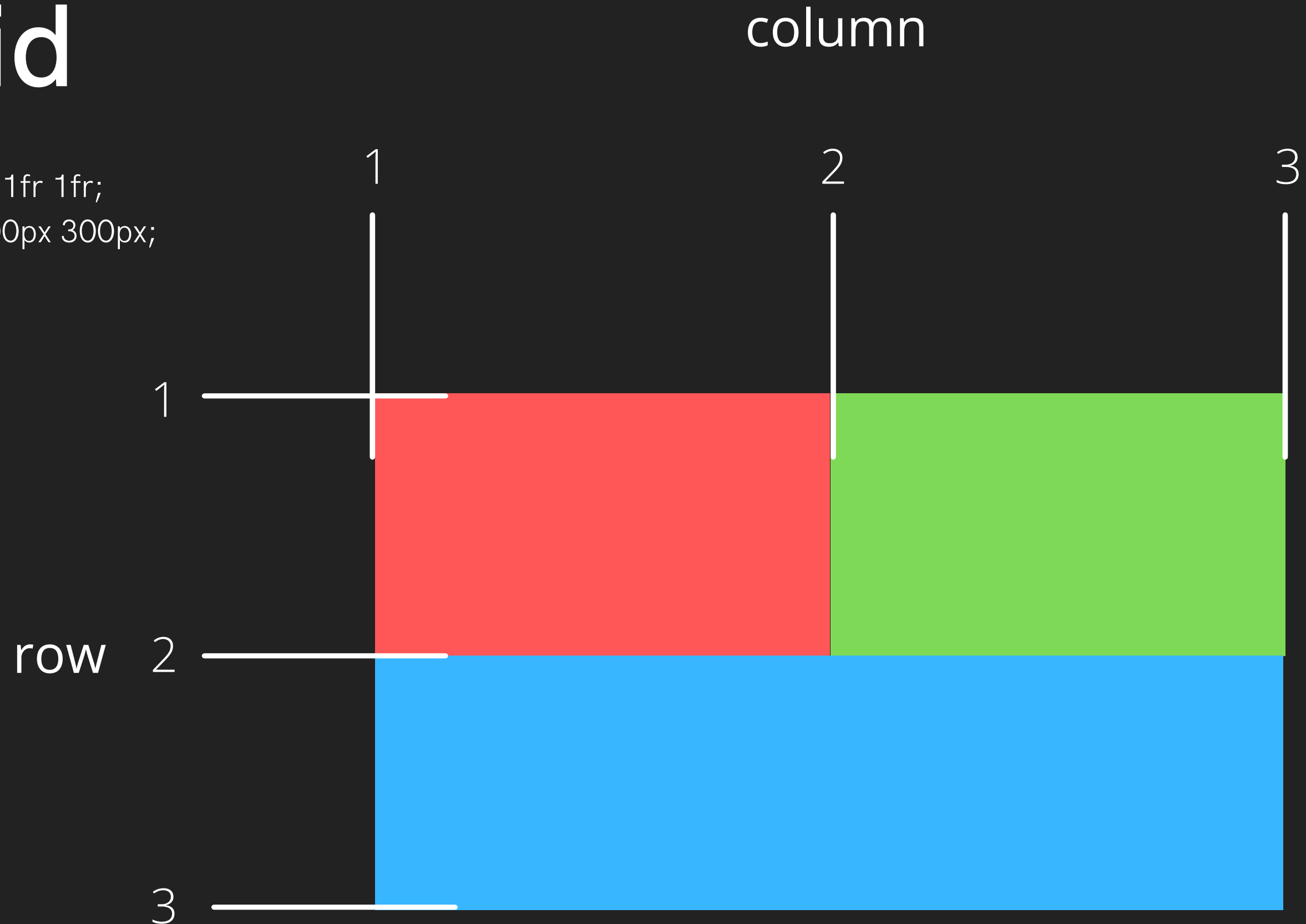
CSS Grid

```
parent {  
  grid-template-column: 1fr 1fr;  
  grid-template-rows: 300px 300px;  
}
```

```
red {  
  grid-column: 1/2;  
  grid-row: 1/2;  
}
```

```
green {  
  grid-column: 2/3;  
  grid-row: 1/2;  
}
```

```
blue {  
  grid-column: 1/3;  
  grid-row: 2/3;  
}
```



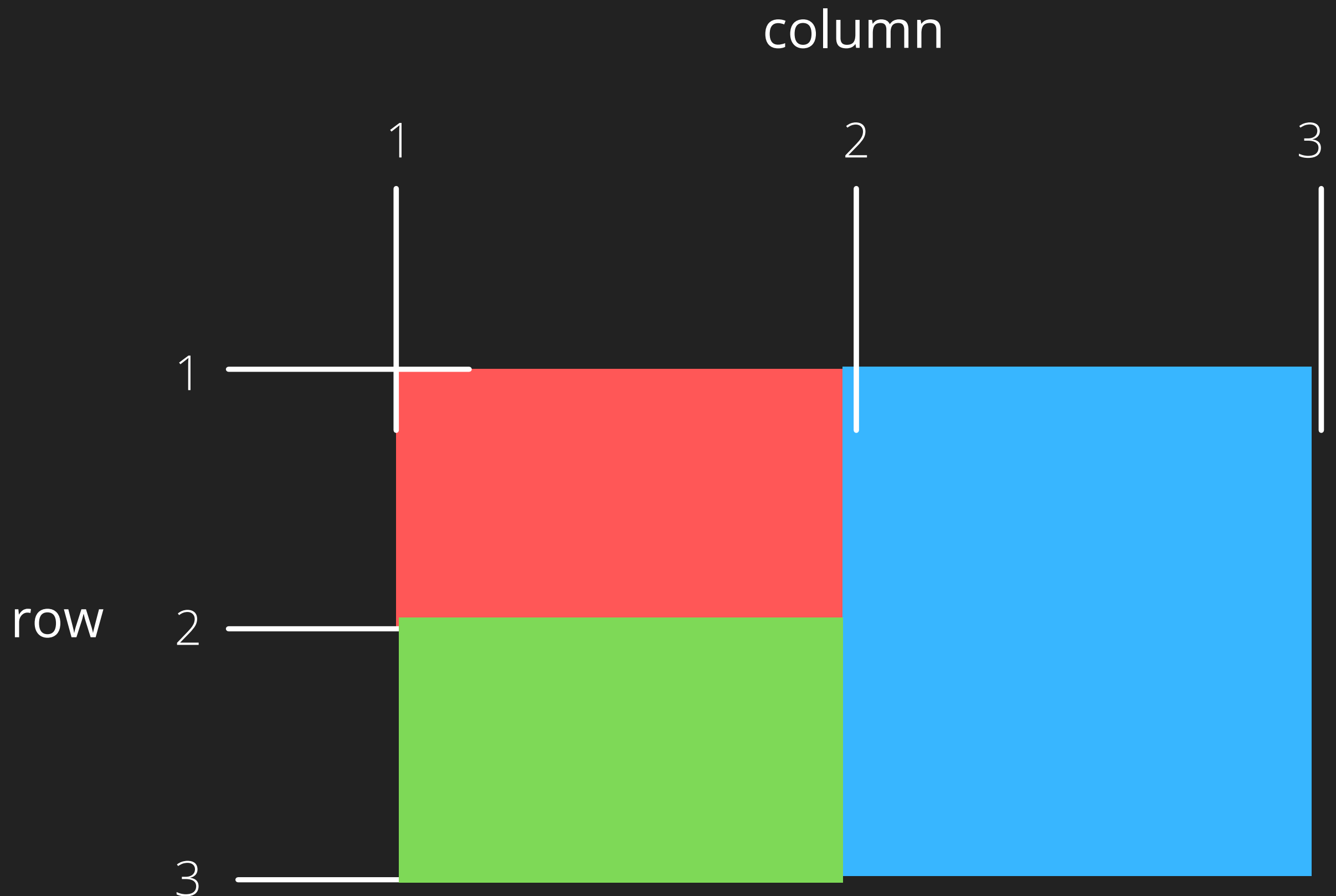
CSS Grid

```
parent {  
  grid-template-column: 1fr 1fr;  
  grid-template-rows: 300px 300px;  
}
```

```
red {  
  grid-column: 1/2;  
  grid-row: 1/2;  
}
```

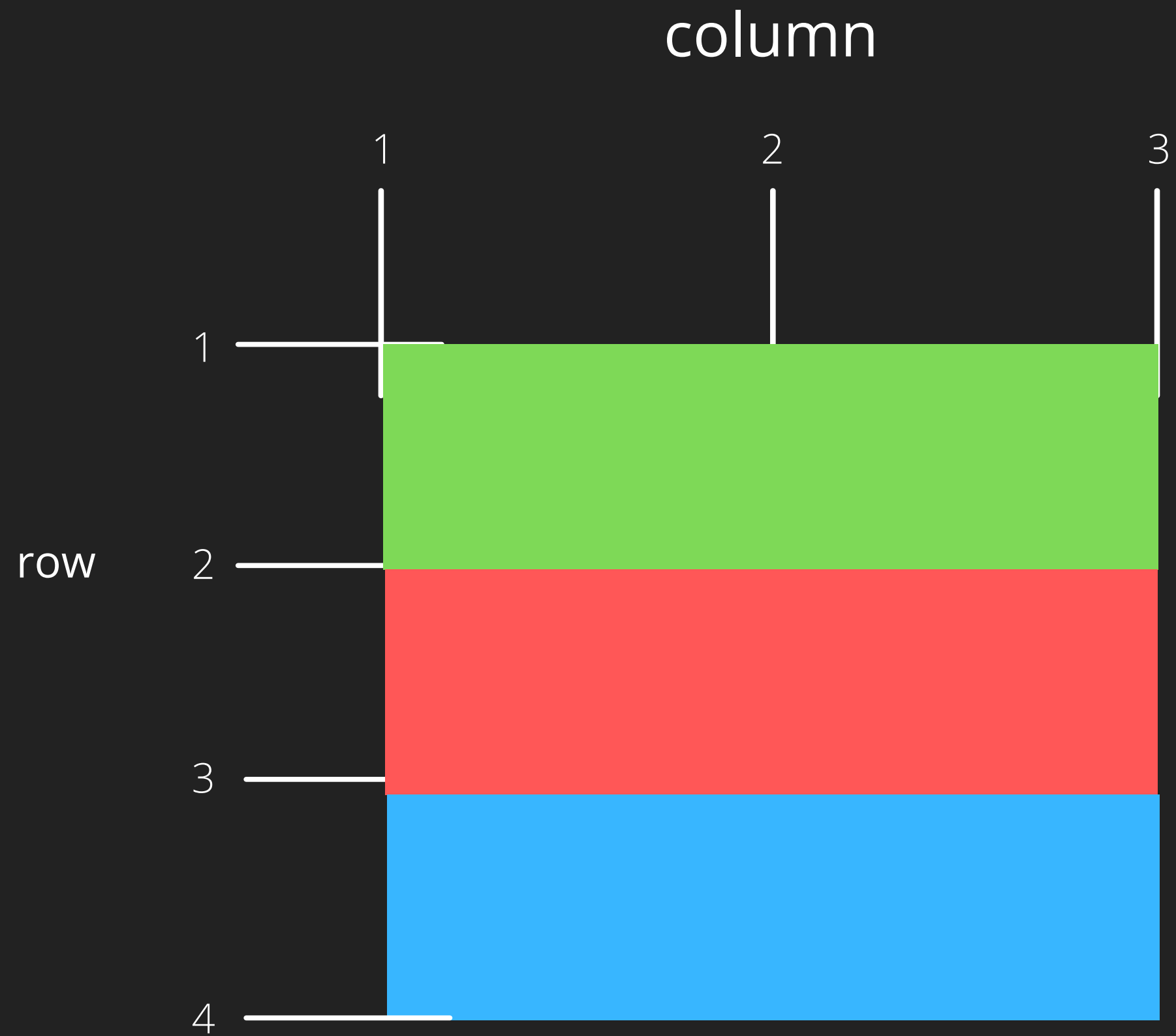
```
green {  
  grid-column: 1/2;  
  grid-row: 2/3;  
}
```

```
blue {  
  grid-column: 2/3;  
  grid-row: 1/3;  
}
```



CSS Grid

CHALLENGE!



Animations

- animation-name
- animation-duration
- animation-delay
- animation-iteration-count
- animation-direction
- animation-timing-function
- **SHORTHAND:** animation

```
@keyframes example {  
  0% {background-color: red;}  
  25% {background-color: yellow;}  
  50% {background-color: blue;}  
  100% {background-color: green;}  
}
```

```
/* The element to apply the animation to */  
div {  
  width: 100px;  
  height: 100px;  
  background-color: red;  
  animation-name: example;  
  animation-duration: 4s;  
}
```

Transitions

- transition-delay
- transition-duration
- transition-property
- transition-timing-function
- SHORTHAND: transition

```
div {  
    transition-property: width;  
    transition-duration: 2s;  
    transition-timing-function: linear;  
    transition-delay: 1s;  
}
```