TDP013 – Web Programming and Interactivity

Lecture 2: HTML, CSS, JavaScript Cookies

Huanyu Li

Human-Centered Systems, Department of Computer and Information Science

LINKÖPING UNIVERSITY

HTML

LINKÖPING UNIVERSITY

HTML and HTML5

- Hypertext Markup Language (HTML)

- 1991, first public description of HTML, “HTML Tags”
- ...
- 2014, HTML5 was published as a W3C recommendation

- It has “Elements” as basic building blocks

HTML Elements

- An element has one start tag and one end tag
  - <START>, </END>
- An element can have attributes
  - listed in start tag of the element
  - value must be quoted
- Content
  - between start tag and end tag

Tag syntax

- Document type

  ```
  <!doctype html>
  ```

- An element with start and end tags

  ```
  <tag>content</tag>
  ```

- An element with attributes

  ```
  <tag attributes>content</tag>
  ```

- An element with attributes and their values

  ```
  <tag attribute_1="value_1" attribute_2="value_2" >content</tag>
  ```

- Self-closing tag

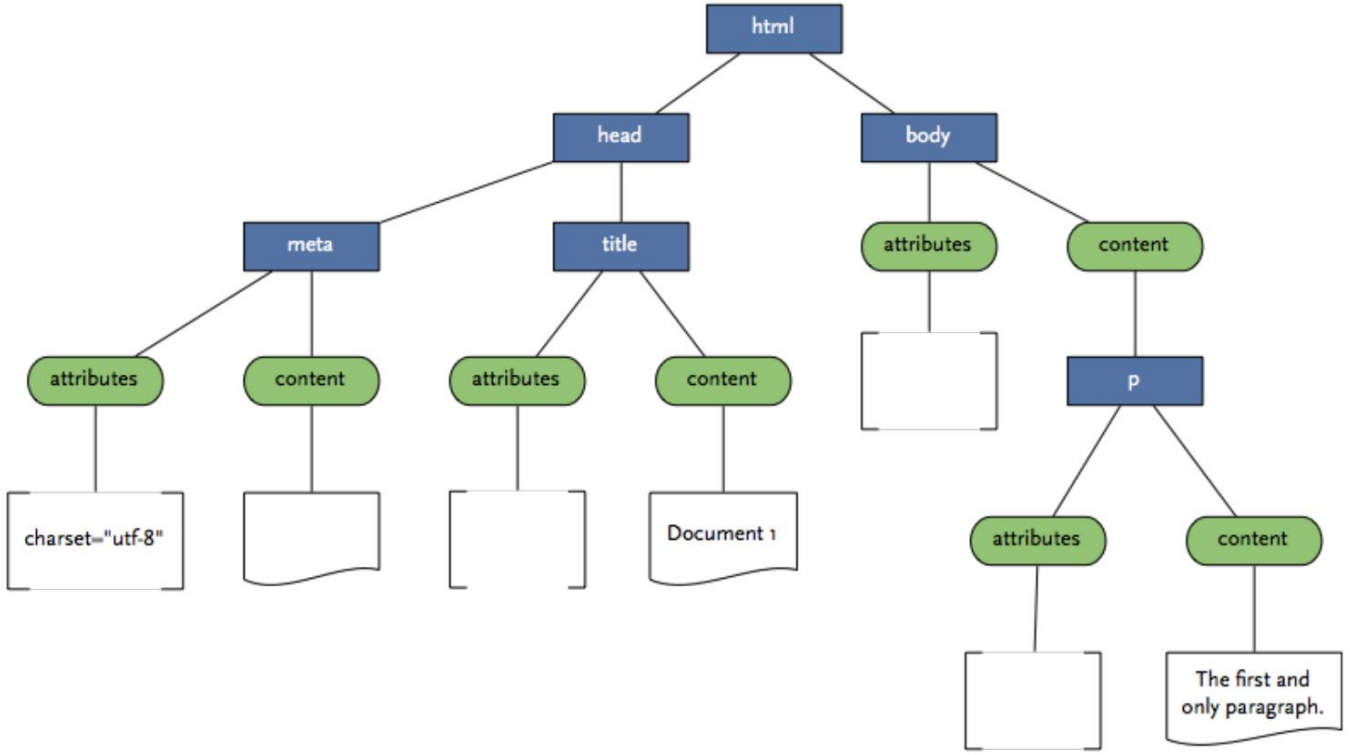
  ```
  <tag/>
  <tag />
  <tag attribute="value" />
  ```

Overall Syntax

- Whitespaces (tab, space, newline) are not significant,
  but are required to make the code structure clear
- Uppercase and lowercase letters do not matter,
  but using lowercase letters for tags is recommended
- Tags form HTML element with nesting elements as a tree

Document Object Model (DOM)

- DOM is a programming API for HTML and XML documents



HTML structure

- We need to organize HTML code for
  - preparing for CSS styles
    - e.g., how the application looks like
  - preparing for interaction
    - e.g., user input
- We need to encode semantics for search engines to use
  - e.g., meaning of tags and attributes

More HTML syntax – Comments

- Comment syntax, disabling HTML code or commenting out code

```
<!—- Commented Content -->
```

or

```
<!—-
<tag>content</tag>
-->
```

More HTML syntax - Headings

- HTML headings at different levels

```
<h1>Heading 1</h1>
<h2>Heading 2</h2>
<h3>Heading 3</h3>
<h4>Heading 4</h4>
<h5>Heading 5</h5>
<h6>Heading 6</h6>
```

More HTML syntax – Paragraphs

- A paragraph always starts with a new line

```
<p>A piece or a paragraph of text</p>

<p>Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do
    eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut
    enim ad minim veniam, quis nostrud exercitation ullamco laboris
    nisi ut aliquip ex ea commodo consequat</p>
```

More HTML syntax – Links, Images

- Hyperlinks will direct you when it is clicked

  ```
  <a href="url">content</a>
  ```

- Image tag does not have a closing tag, it has two required attributes
  - src, specifies the path to the image
  - alt, specifies an alternate text for the image

  ```
  <img src="./example.jpg" alt="Example image.">
  ```

More HTML syntax – Structural elements

- Block Elements (paragraphs and sections/divisions)
  - always start on a new line and take up full available width
  - are used to organize a page into different blocks
  - <p> element and element

- Inline Element,
  - does not start in a new line, it takes up width as necessary

```
<p>paragraph1</p>
<p>paragraph2</p>

<span>span1</span>
<span>span2</span>
```

Well formed HTML documents

- A valid HTML document
  - e.g., correct nesting, start tag has corresponding end tag
- Browsers do not need to guess how to interpret the code
- HTML validator: https://validator.w3.org/

- In the lab, the code should be validated as HTML5
  - correct one identified error at a time until it passes when using a validator
  - sometimes validators complain about strange things. If you feel convinced that you are right and the validator is wrong, that is okay.

CSS
syntax and structure

LINKÖPING UNIVERSITY

CSS (Cascading Style Sheets)

- CSS specifies
  - visual style
  - layout
- CSS does not specify HTML structure
- HTML elements inherit CSS properties from their parent elements
  - e.g., font, color, visibility

CSS styling

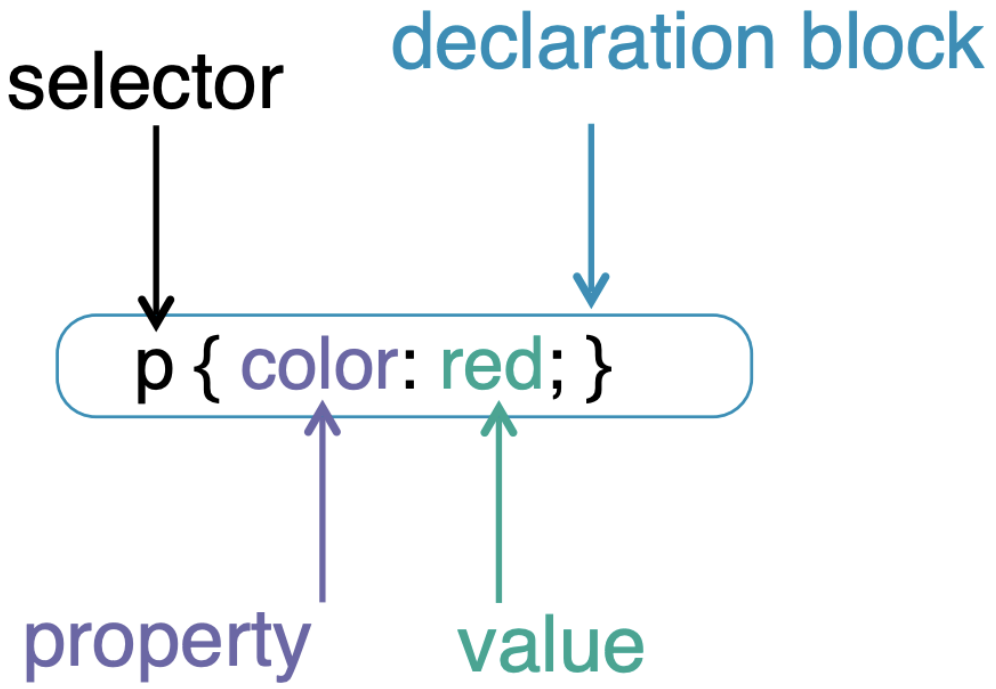
- Colors and borders
  - color of the background of an element (headings, paragraphs)
  - color of the border of an element (headings, paragraphs)
  - text color (text content in headings, paragraphs)
- Typography
  - font, font size, line height
- Element Size
  - height, width

CSS definition

- A text file with the extension .css
- Contains a collection of style specifications
- An HTML document can link to one or more stylesheets
- Style specifications are evaluated in order (top to bottom) and by specificity
- Later specifications can run over earlier specifications

CSS Syntax

- selector
  - which HTML element to style
- declaration block
  - contains one or more declarations separated by semicolons
  - each includes a CSS property name and a value, separated by colon
- property
- value



CSS Syntax

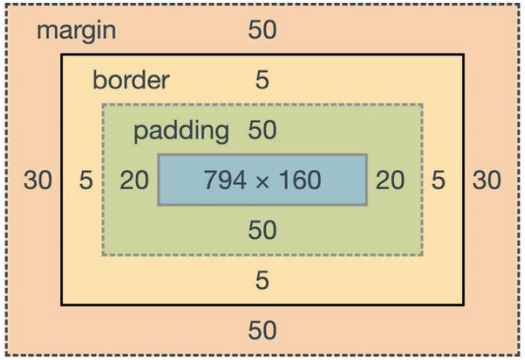
```
/* This is a comment */
body {
        font-family: Times, Serif;
        font-size: 16px;
}
h1 {
        font-family: Helvetica, Sans-Serif;
        font-size: 32px;
}
```

CSS Properties

- Display: display behaviour of an element
  - *{display: value;}*
- Backgrounds: area behind an element's content, color, image, position, etc.
  - *background-color, background-image, etc.*
- Borders: visual outline around an element
  - *border-style, border-width, border-color, border-radius*
- Font
  - *font-family, font-size, font-weight, etc.*
- Colors (background, text, border)
  - *{background-color: value}, {color: value}, {border: value}*

https://www.geeksforgeeks.org/css/css-tutorial/

CSS Properties



- Margins: space around an element
  - *{margin: value;}, {margin-top: value;}, {margin-right: value;},*
  - *{margin-left: value;}, {margin-bottom: value;}*
- Padding: space between the content and defined border of an element
  - *{padding: value;}, {padding-top: value;}, {padding-right: value;},*
  - *{padding-left: value;}, {padding-bottom: value;}*
- Position: how to place elements (layout, stacking order, alignment)
  - *{position: relative; top: 10px; left: 20px}*

https://www.geeksforgeeks.org/css/css-tutorial/

Code Example

CSS Units of measure

- px: pixels on the screen (but a CSS pixel is not quite the same as a screen pixel)
- em: inherited font-size
  - 1em = inherited font-size
  - 2em = double inherited font-size
- rem: like em but inherited from the root element
- pt: point (1pt = 1/72 inch)
- %: percentage of inherited size
- vw: percentage of the window (viewport) width
- vh: percentage of the window (viewport) height

HTML + CSS

- HTML code can link to CSS files
- CSS code can be written directly in HTML code,
  - but in the labs, they should be written in external files
- The link from HTML to CSS is made using <link> in the header tag

HTML + CSS

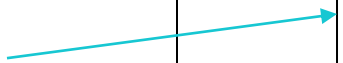
page.html

```
<!doctype html>
<html lang="en">

<head>
        <meta charset="utf-8">
        <title>A document</title>
        <link rel="stylesheet" href="style.css">
</head>

<body>
<h1>The Heading</h1>
<p>The paragraph</p>
</body>

</html>
```



style.css

```
/* This is a comment */
body {
        font-family: Times, Serif;
        font-size: 16px;
        padding: 0px 0px 0px 0px;
}
h1 {
        font-family: Helvetica, Sans-Serif;
        font-size: 32px;
}
```

File paths in HTML and CSS

- File paths relative to current file
- ./ points to the same level in the file structure
- ../ points to one level up in the file structure
- / refers to the root of the website

- File paths can also be absolute

```
<!doctype html>
<html lang="en">

<head>
        <meta charset="utf-8">
        <title>A document</title>
        <link rel="stylesheet" href="https://example.com/style.c
</head>

…

</html>
```

Some common style properties

- font-size: font size
- font-family: font name
- line-height: line height
- width: element width
- height: element height
- margin: distance to the next element
- padding: distance from the element's edge to the content
- color: font color
- background-color: background color of the element
- border: border-style of the element

Style properties can take several arguments

```
/* This is a comment */
body {
        font-family: Times, Serif;
        font-size: 16px;
        padding: 0px 0px 0px 0px; /* same as padding: 0px */

}
h1 {
        font-family: Helvetica, Sans-Serif;
        font-size: 32px;
}
```

CSS-reset

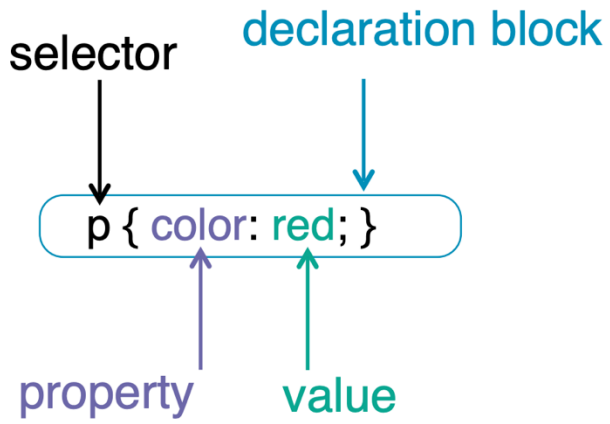
- Browsers always have predefined stylesheets by default
- Some things differ between browsers (e.g. text fonts, heading sizes, etc.)
- CSS reset: CSS code that aims to reset all properties to known values. Creates a well-defined foundation to build on
- Popular variants
  - Eric Meyer's CSS reset from 2011
    - https://meyerweb.com/eric/tools/css/reset/reset.css
  - Normalize.css
    - http://nicolasgallagher.com/about-normalize-css/
  - Reboot, Resets, and Reasoning
    - https://css-tricks.com/reboot-resets-reasoning/

CSS
selectors

LINKÖPING UNIVERSITY

CSS-Selectors

- Select a group of elements
  - “select all paragraphs and headings”
- Select adjacent sibling elements at the same level
  - “select all paragraphs followed by a heading”
- Select all descendants
  - “select all images contained within a div tag”
- Select all children
  - “select all list elements in lists within a given class”



Selectors – group of elements

```
/* Target all h1, h2 and h3 elements */

h1, h2, h3 {
  border: 2px solid black;
}
```

Selectors – Descendant combination

```
/* Select all li element that are nested within a nav element */
nav li {
  color: #F00;
}
```

Selectors – Child combination

```
/* Target all p elements that are children of a div */
div > p {
  border: 2px solid #000;
}
```

Selectors – Adjacent siblings

```
/* Target all p elements that follow a h1 */
h1 + p {
  font-weight: bold;
}
```

Selectors – pseudo selectors

- :hover
  - Select the element that the mouse hovers over. Ex. change the appearance of a link when the mouse is over it
  - CSS: a:hover { … }
  - HTML: Try hovering over this link.
- :visited
  - Select elements (i.e. links) that have been visited previously a:visited

Selectors – First child

```
/* Target the first child of every div element */

div:first-child {
  font-weight: bold;
}
```

Selectors – Last child

```
/* Target the last child of every div element */

div:last-child {
  font-weight: bold;
}
```

Selectors – :nth child

```
/* Target the 3rd child of every div element */
div:nth-child(3) {
  font-weight: bold;
}
/* Target the odd numbered children of every div element */
div:nth-child(odd) {
  font-weight: bold;
}
/* Target the even numbered children of every div element */
div:nth-child(even) {
  font-weight: bold;
}
/* Target every 3rd child of every div element */
div:nth-child(3n) {
  font-weight: bold;
}
```

What can't you do with CSS?

- Select a parent element of an element
- Select from the root upwards (i.e. you can't reverse the order)
- If we want to handle that kind of styling, we have to use JavaScript!

CSS
Class and ID

LINKÖPING UNIVERSITY

What is a class?

- Elements can be associated with one or more classes
- More than one element can have the same class
- Classes can be used for styling recurring components
- The prefix . (dot) is used before class names in CSS selectors
- Example:

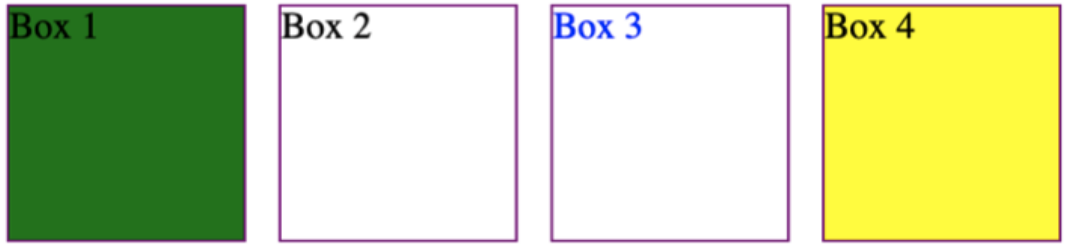
```
<div class="box">
<div class="box green-bg">
```

.box, div.box, div.box.green-bg, .green-bg

Example with several classes

```css
.green-bg {
  background-color: green;
}
.yellow-bg {
  background-color: yellow;
}
.blue-fg {
  color: blue;
}
.box {
  border: 1px solid purple;
  width: 100px;
  height: 100px;
  margin: 5px;
  display: inline-block;
}
```

```html
<div>
        <div class="green-bg box">Box 1</div>
        <div class="green-yellow box">Box 2</div>
        <div class="box blue-fg">Box 3</div>
        <div class="green-bg yellow-bg box">Box 4</div>
</div>
```



What is an ID?

- An element can have exactly one ID
- All IDs must be unique in the HTML document
- IDs can be used to add a style to a specific element
- The prefix # is used before IDs in CSS selectors
- Example:

```
<div id="footer">
```

#footer, div#footer, h1#main-heading, #main-heading

Example in CSS

```
.infobox {
  font-family: Helvetica, Arial, Sans-Serif;
  font-size: 0.9em;
  background-color: #999;
  color: #000;
  border: 2px solid black;
}
#menu {
  background-color: #000;
  color: #FFF;
}
```

CSS
Colors and borders

LINKÖPING UNIVERSITY

Defining Colors with RGB

- RGB (red - green - blue) is an additive color model
- Values from 0-255 (decimal) or 0-F or 00-FF (hex)
  - Black rgb(0, 0, 0) or #000 or #000000
  - White rgb(255, 255, 255) or #FFF or #FFFFFF
  - Purple rgb(128, 0, 255) or #8000FF
- Named colors: black, silver, gray, white, maroon, red, purple, fuchsia, green, lime, olive, yellow, navy, blue, ...

background-color, color

```
h1 {
  color: #fff;
  background-color: #075488;
}
```

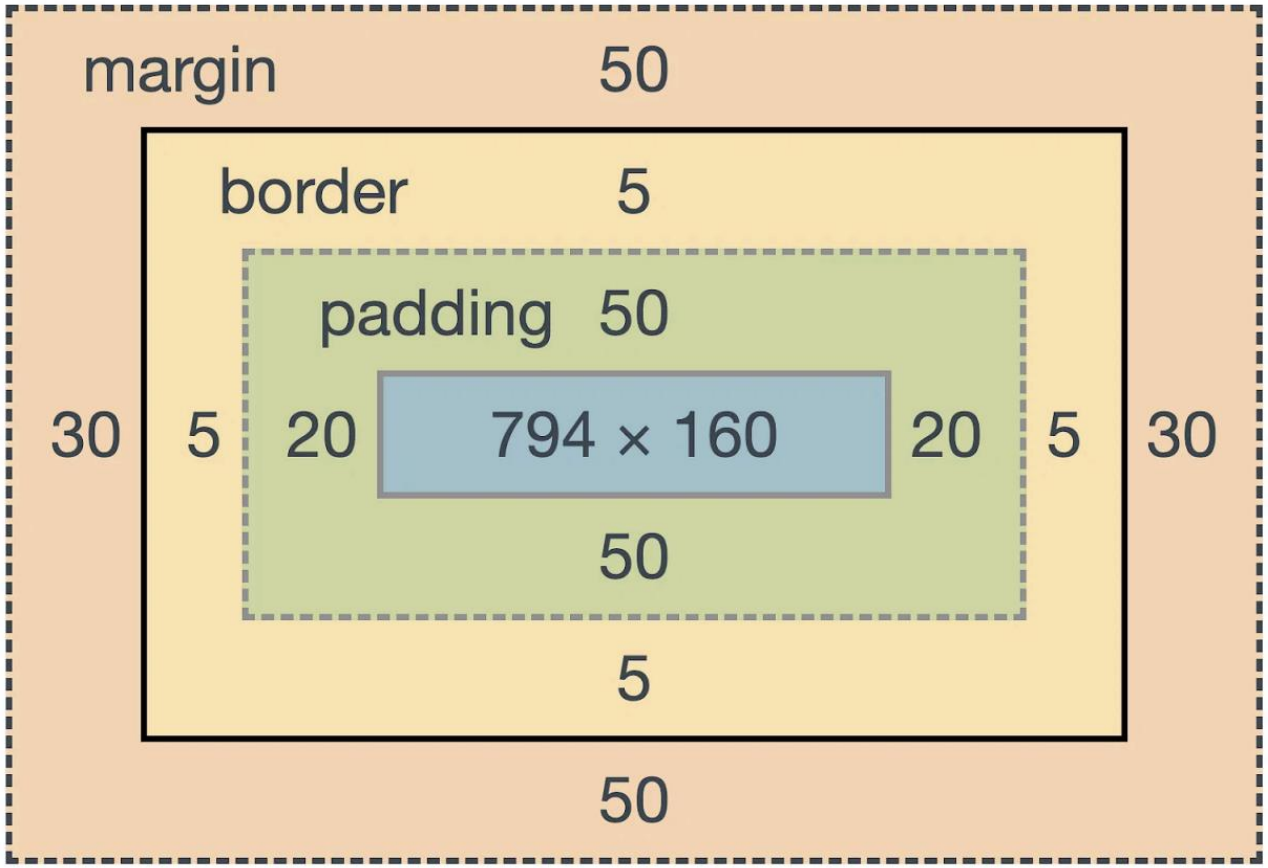
CSS borders

- Some types of elements can have a border
- A border has the following properties:
  - width: the width of the line
  - style: the type of line (for example, dashed)
  - color: the color of the line
- Each property can be defined for each side (top, right, bottom or left)
- The corners can also be rounded with border-radius

CSS
Box models

LINKÖPING UNIVERSITY

Box model



Picture: https://www.geeksforgeeks.org/css/css-box-model/

Definition of padding

- *{padding: top_value right_value bottom_value left_value;}*
- *{padding-top: value;}*
- *{padding-right: value;},*
- *{padding-left: value;}*
- *{padding-bottom: value;}*

Definition of margin

- *{margin: top_value right_value bottom_value left_value;}*
- *{margin-top: value;}*
- *{margin-right: value;},*
- *{margin-left: value;}*
- *{margin-bottom: value;}*

Code Example

CSS
Layout model

LINKÖPING UNIVERSITY

What should you learn?

- Every element exists inside a layout context
- Every element has a layout context
- Flexbox and grid layouts are useful (but can't solve everything!)
- Good to know how normal flow + positioning works

Layout models in CSS

- Normal flow + positioning
  - display: block | inline | inline-block
- Flexbox
  - display: flex
- Grid layout
  - display: grid
- Floats
  - float: left | right
- Table layout
- Multiple-column layout

Normal flow

- Elements can behave differently depending on the display type:
  - inline
  - block
  - inline-block
- inline means that the element is placed together with the text. The generic inline element is
- block means that the element is outside the text. The generic block element is . A block element extends across the entire available width unless otherwise specified
- inline-block is a block that can be placed together with text.

display property

```
/* The formatting context is set using
the display property */
.infobox {
  display: block;
}

.question {
  display: inline;
}
```

Five position models

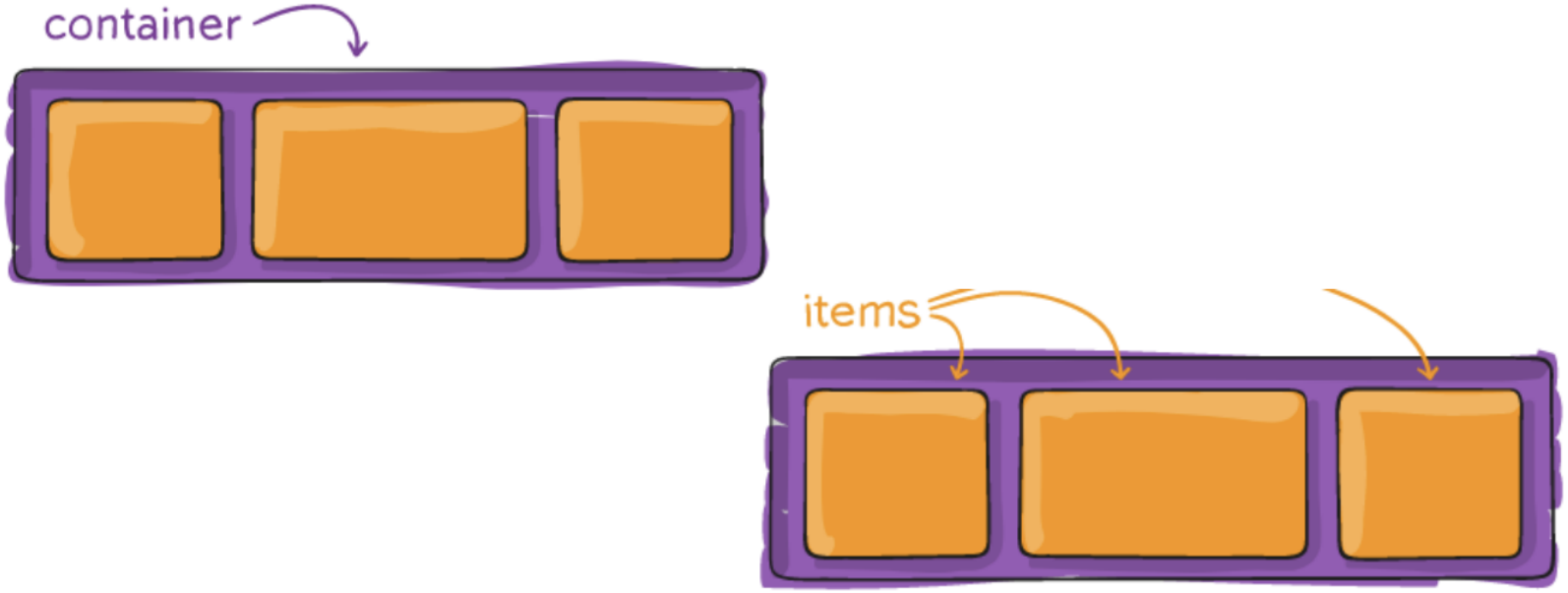
- Position determines how a block is positioned on the page in relation to others
  - **static** (default): positions elements in the order they appear in the document. The top, right, bottom, left and z-index properties have no effect
  - **relative**: similar to static but allows the top, right, bottom, left and z-index properties to be used to change the position relative to the previous element.
  - **absolute**: positions the element in relation to the nearest non-static parent. The top, right, bottom, left and z-index properties are used to change the position.
  - **fixed**: behaves the same as absolute but in relation to the browser window (viewport)
  - **sticky**: behaves as relative until a specified offset is met after which it behaves as fixed.
- Example: https://jsfiddle.net/4nepm12L/46/

CSS
Flexbox Layout

LINKÖPING UNIVERSITY

Flexbox layout

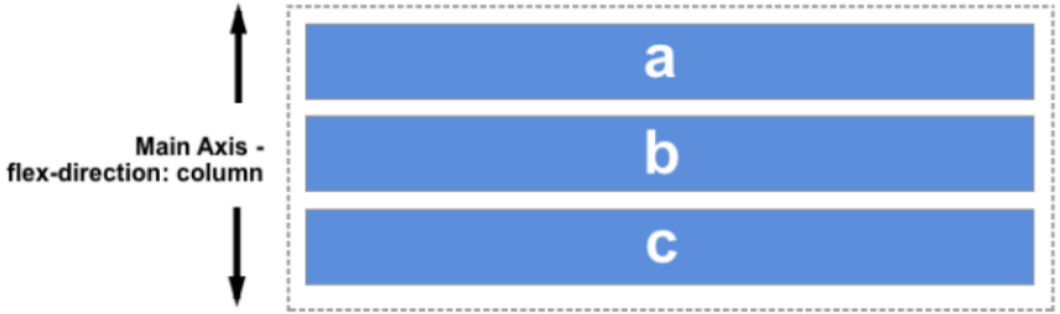
- Multiple properties are used together
- Some properties are set at the container level (flex container)
- Some properties are set on the elements in the container (flex items)



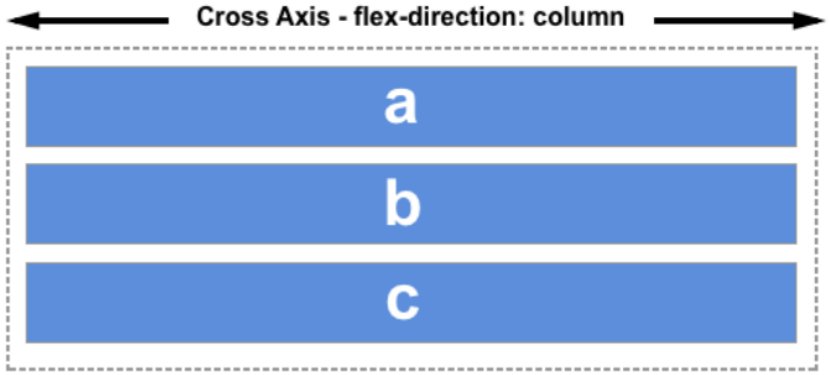
Pictures: https://css-tricks.com/snippets/css/a-guide-to-flexbox/

Main axis and crossing axis

Choose `column` or `column-reverse` and your main axis will run top to bottom of the page in the block direction.



If your main axis is `column` or `column-reverse` then the cross axis runs along the rows.



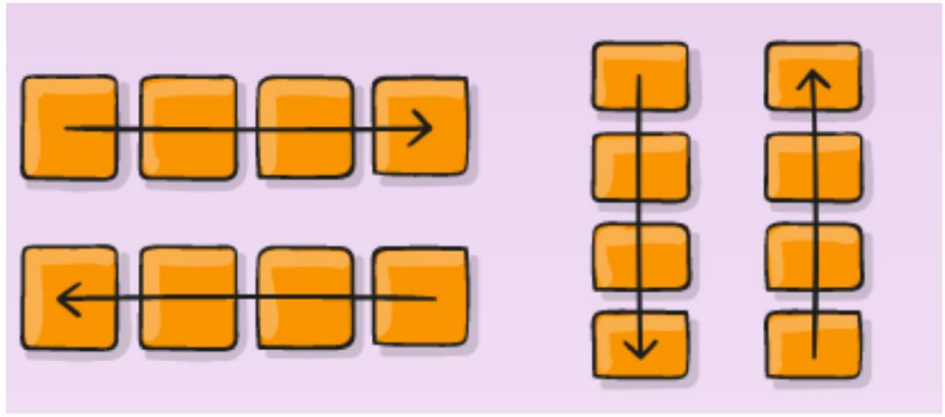
Pictures: https://developer.mozilla.org

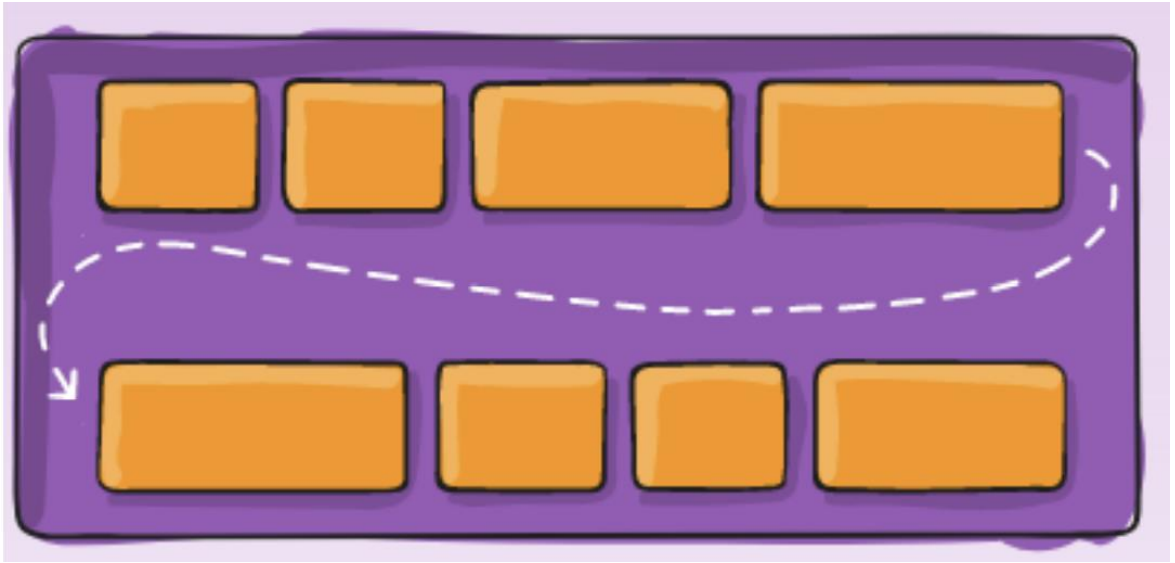
Some properties at the container level

- display: flex
- flex-direction: row | row-reverse | column | column-reverse
- flex-wrap: nowrap | wrap | wrap-reverse
- justify-content: flex-start | flex-end | center | space-between
- align-items: stretch | flex-start | flex-end | center | baseline
- flex-flow: <flex-direction> <flex-wrap>

flex-direction and flex-wrap

- flex-direction: row | row-reverse | column | column-reverse
- flex-wrap: nowrap | wrap | wrap-reverse

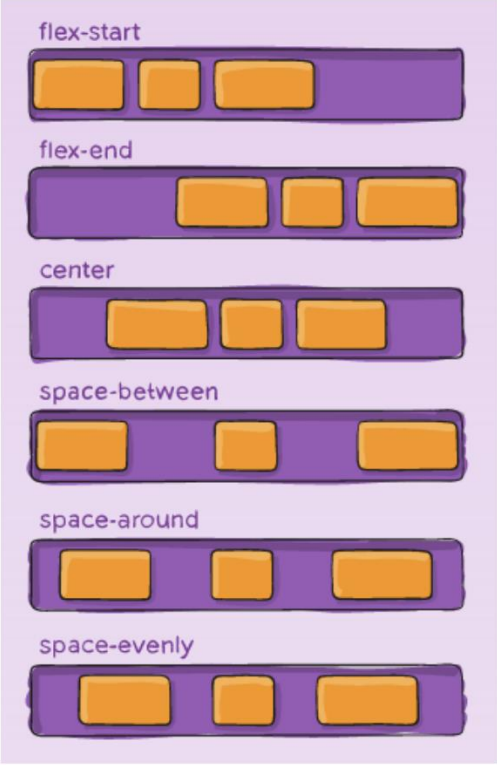




Pictures: https://css-tricks.com/snippets/css/a-guide-to-flexbox/

justify-content

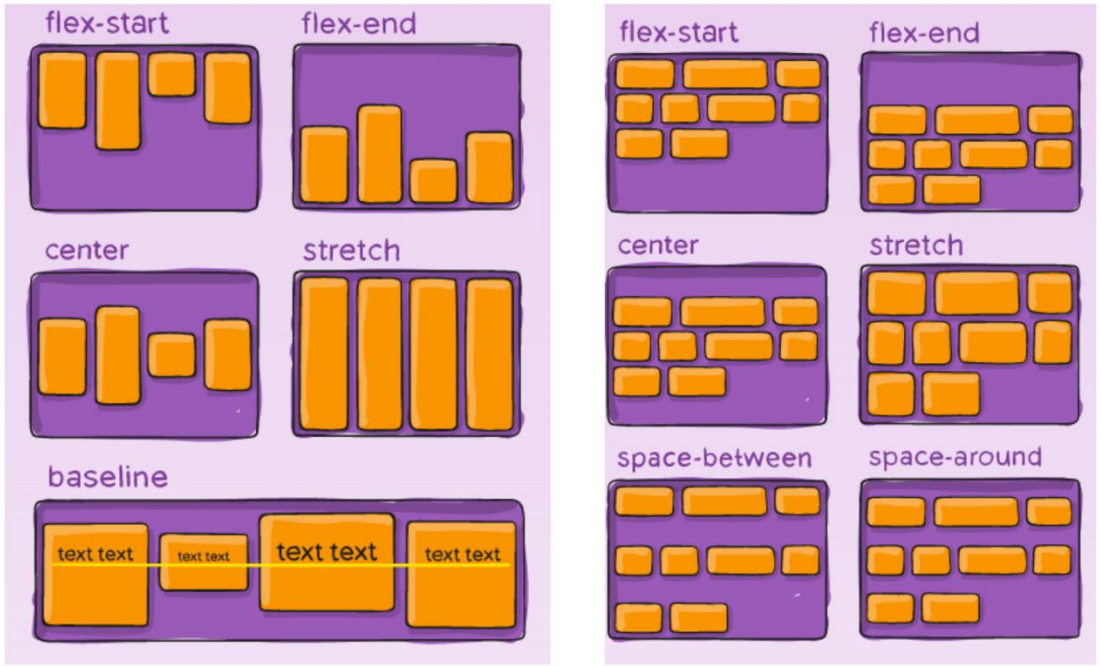
- justify-content: flex-start | flex-end | center | space-between



Pictures: https://css-tricks.com/snippets/css/a-guide-to-flexbox/

align-items and align-content

- align-items: stretch | flex-start | flex-end | center | baseline
- align-content: stretch | flex-start | flex-end | center | space-between | space around



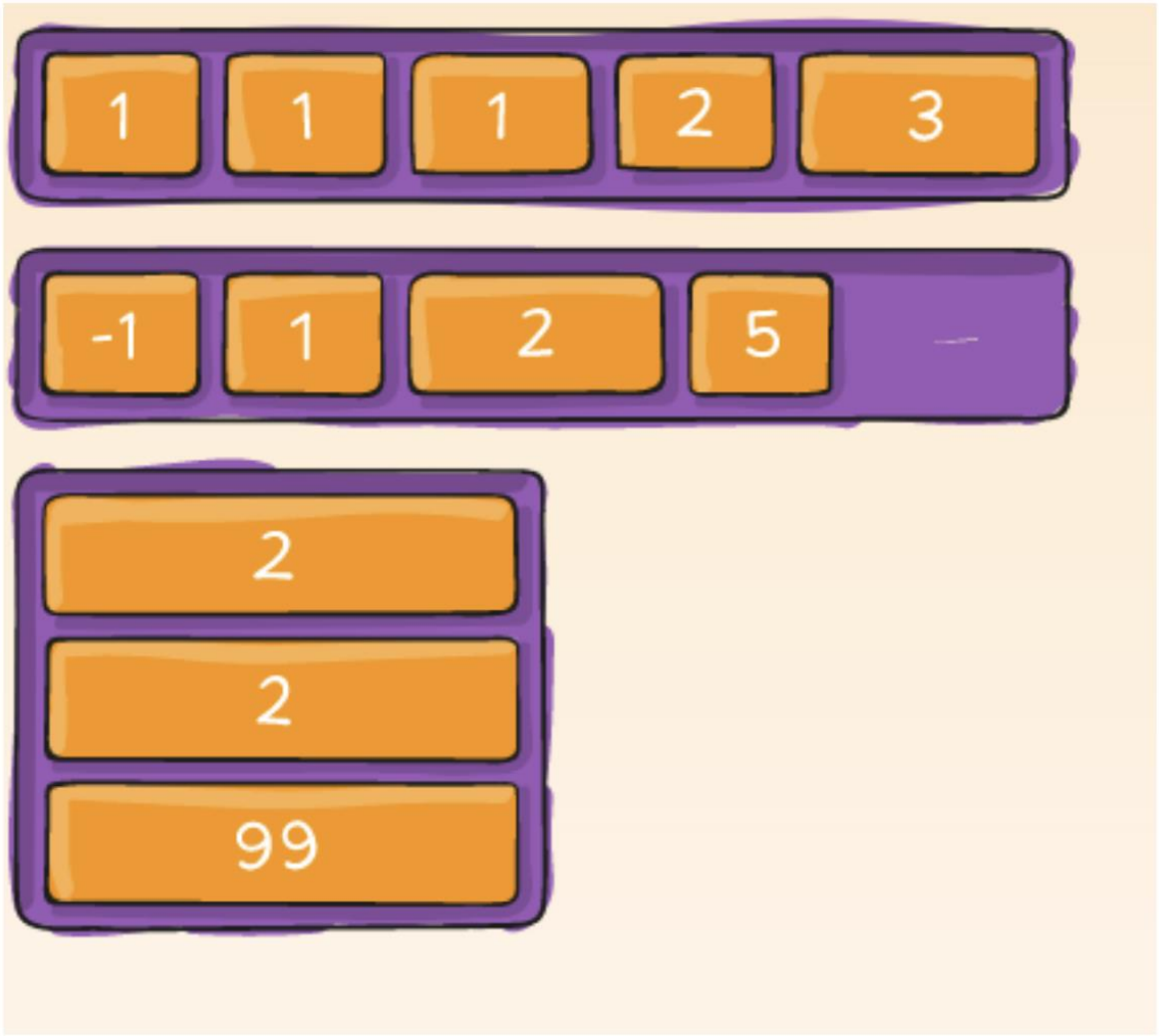
Pictures: https://css-tricks.com/snippets/css/a-guide-to-flexbox/

properties for flex items

- order: <integer>
- flex-grow: <positive number>
- flex-shrink: <positive number>
- flex-basis: <width/height>
- align-self: auto | flex-start | flex-end | center | baseline | stretch;
- flex

order

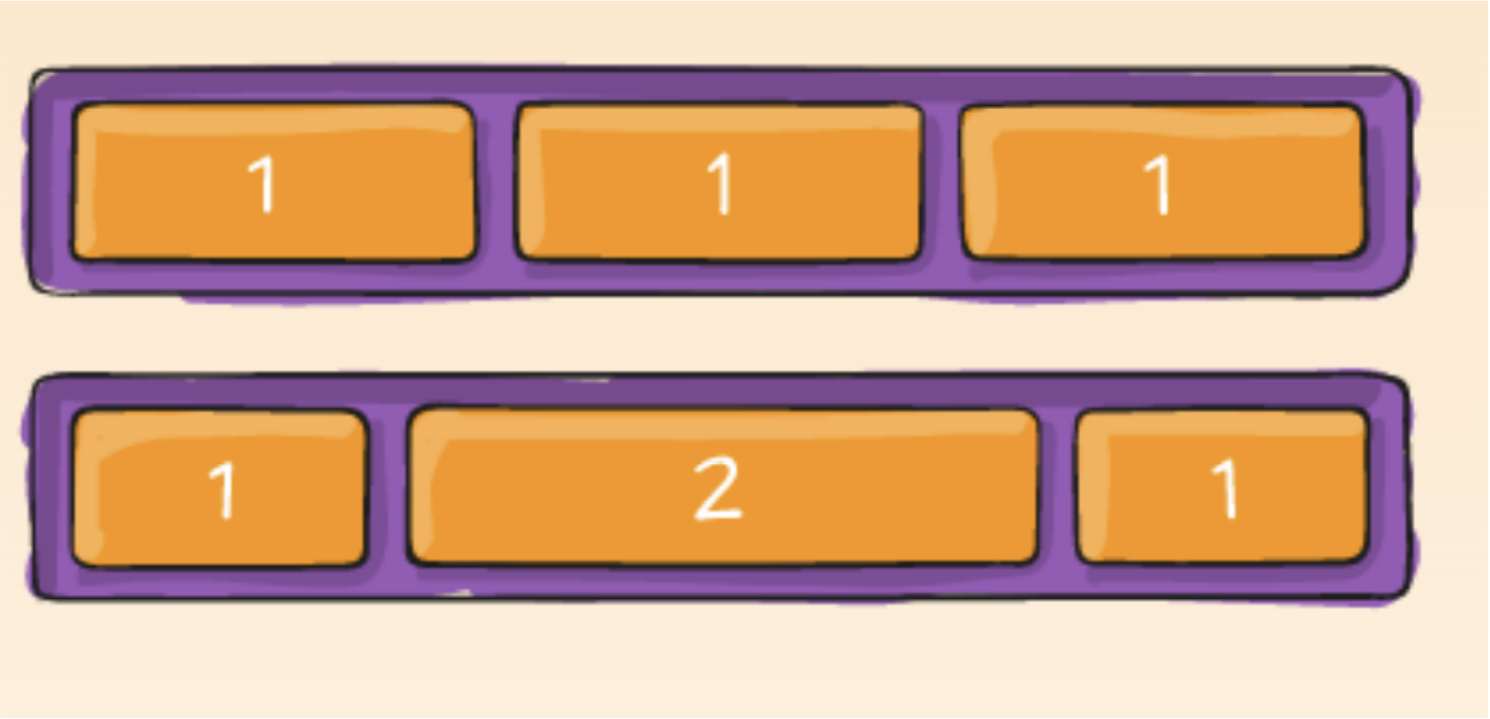
- order: <integer>



Pictures: https://css-tricks.com/snippets/css/a-guide-to-flexbox/

flex-grow and flex-shrink

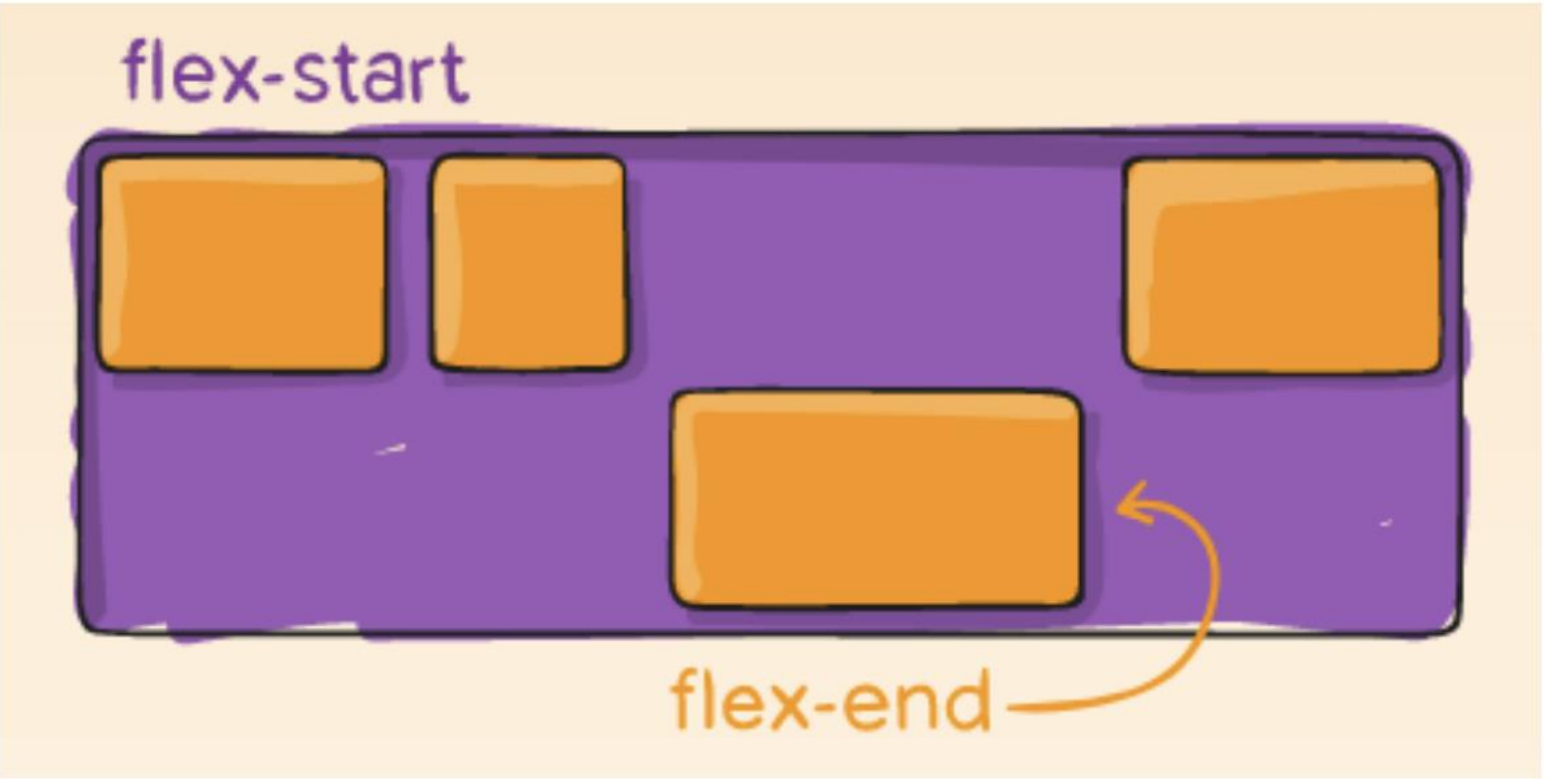
- flex-grow: <positive number>
- flex-shrink: <positive number>



Pictures: https://css-tricks.com/snippets/css/a-guide-to-flexbox/

align-self

- align-self: auto | flex-start | flex-end | center | baseline | stretch;



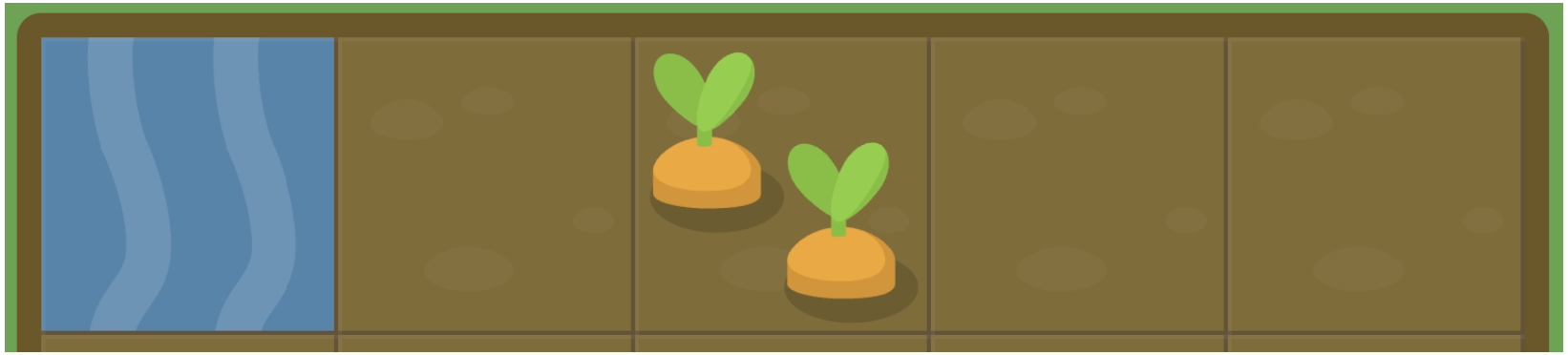
Pictures: https://css-tricks.com/snippets/css/a-guide-to-flexbox/

Flexbox Froggy

https://flexboxfroggy.com/

Grid Garden

https://cssgridgarden.com/



JavaScript Cookies

Cookies

- Small pieces of data stored in key-value pairs on a visitor's machine, used to remember information about the visitor
  - "website places on visitor's browsers"
  - they allow websites you visit to recognize your machine when you re-visit
- Companies can use cookies to customize

- Cookie operations
  - create, read, update, delete

JavaScript Cookies

```
// Set a cookie
document.cookie = "username=John Smith; expires=Thu, 18 Dec 2025 12:00:00 UTC;
path=/";


// Read cookies
console.log(document.cookie); // "username=John Smith"


// Delete a cookie (set it to expire in the past)
document.cookie = "username=John Smith; expires=Thu, 18 Dec 2024 12:00:00 UTC;
path=/";
```

LINKÖPING UNIVERSITY