Introduction

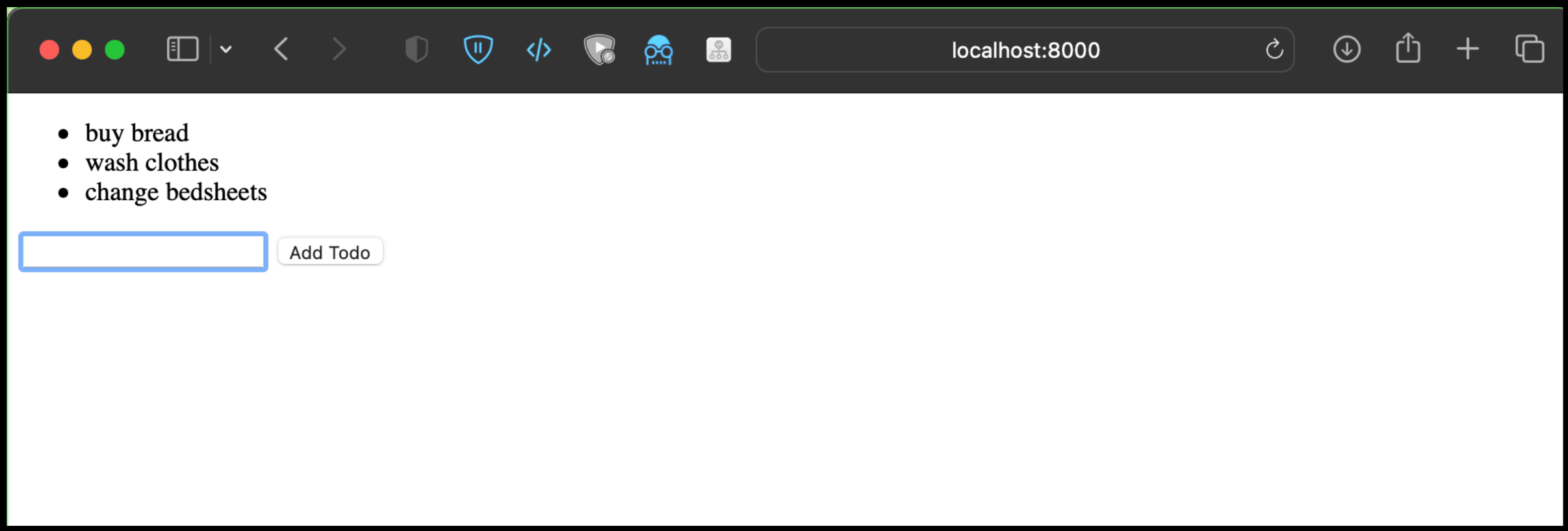
overview and recap of frontend web development

Day 1 - Fundamentals of Frontend Development

Which one will make you happy?

Vanilla

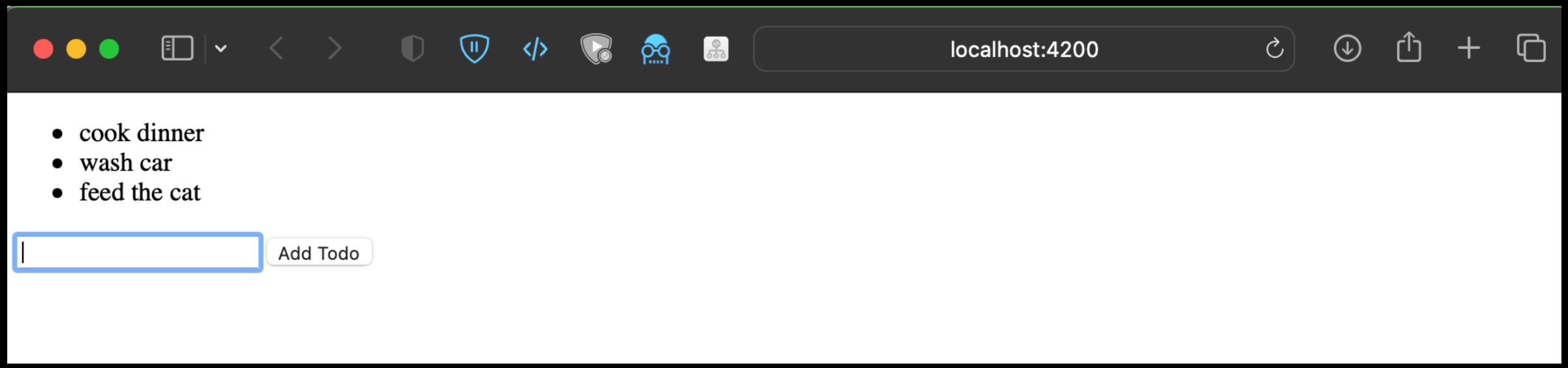
For the brave only




```
npm install http-server -g
npx http-server
```

Angular

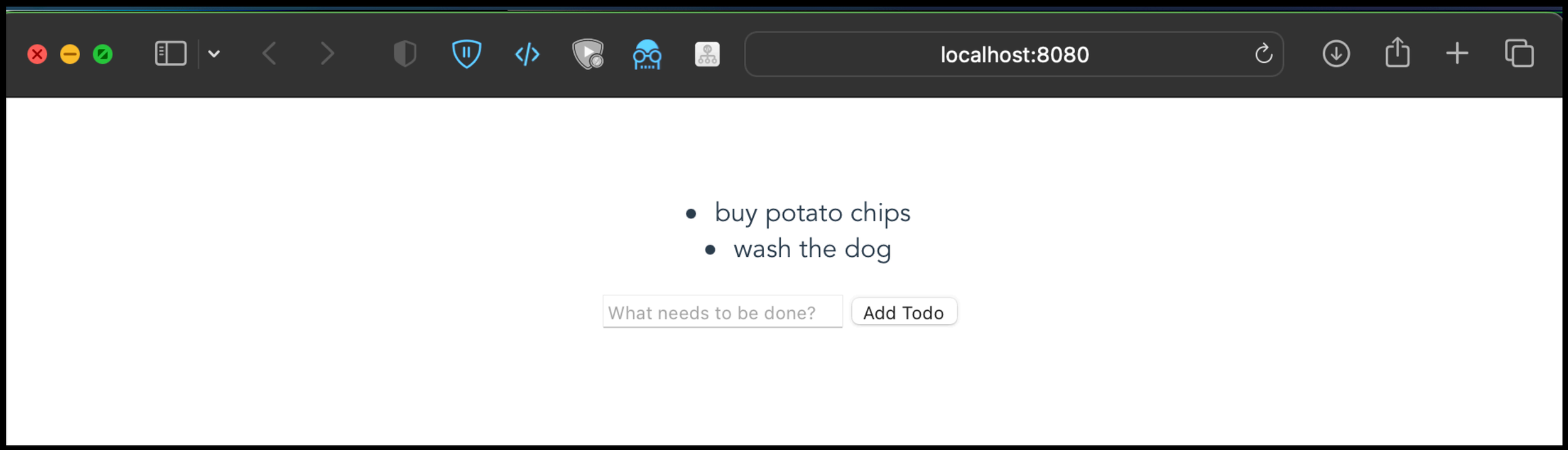
Very mature




```
npm install -g @angular/cli
ng new [PROJECT NAME]
cd [PROJECT NAME]
ng serve
```

Vue.js

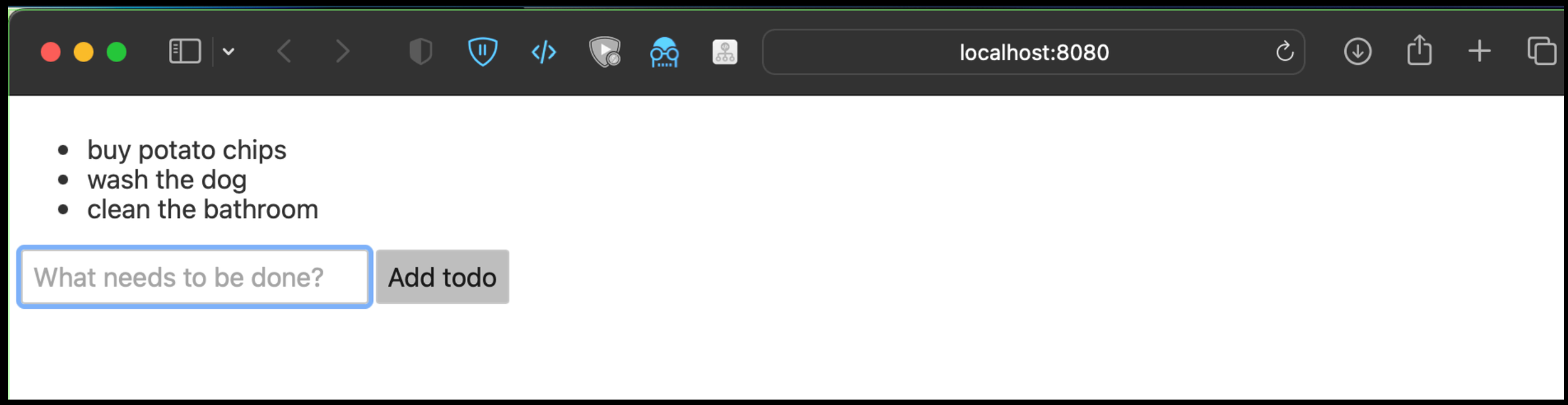
Tiny and fast




```
npm install -g @vue/cli
vue --version
vue create vue-app
```

Svelte

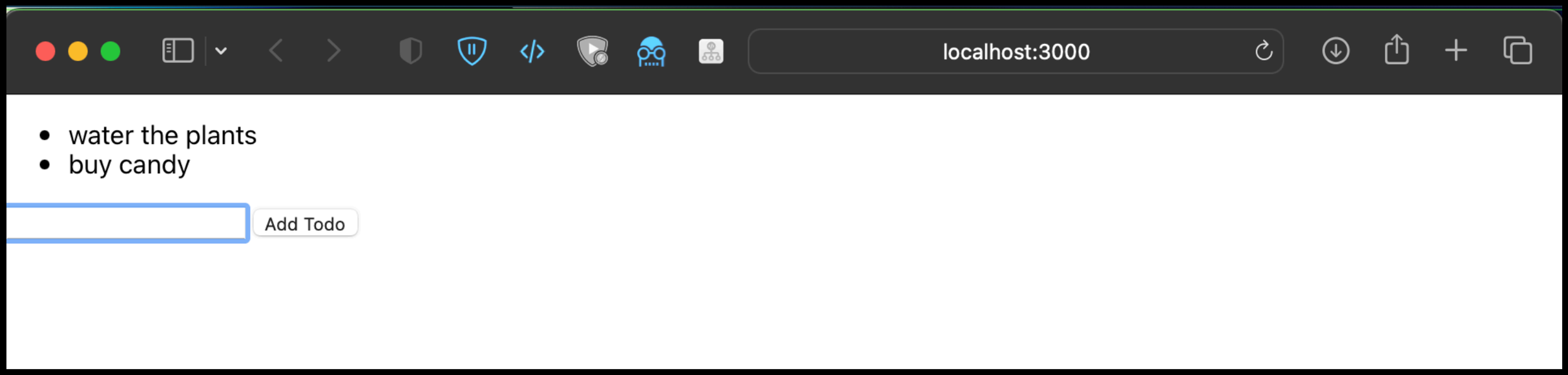
No virtual DOM




```
npx degit sveltejs/template svelte-app
cd svelte-app
npm install
```

Solidjs

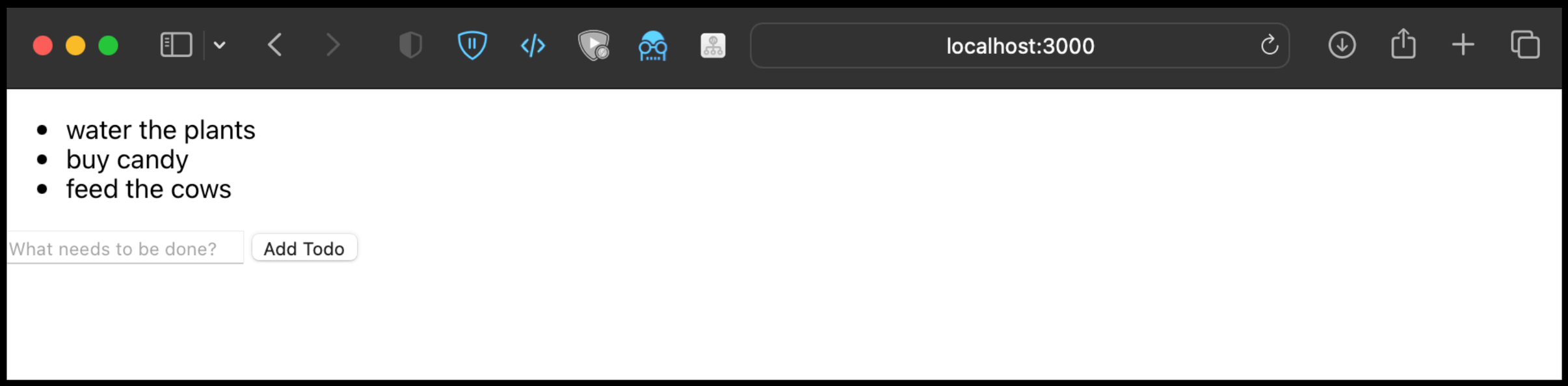
No virtual DOM, fast




```
npx degit solidjs/templates/js solid-app
cd solid-app
npm install
```

React

Simply popular

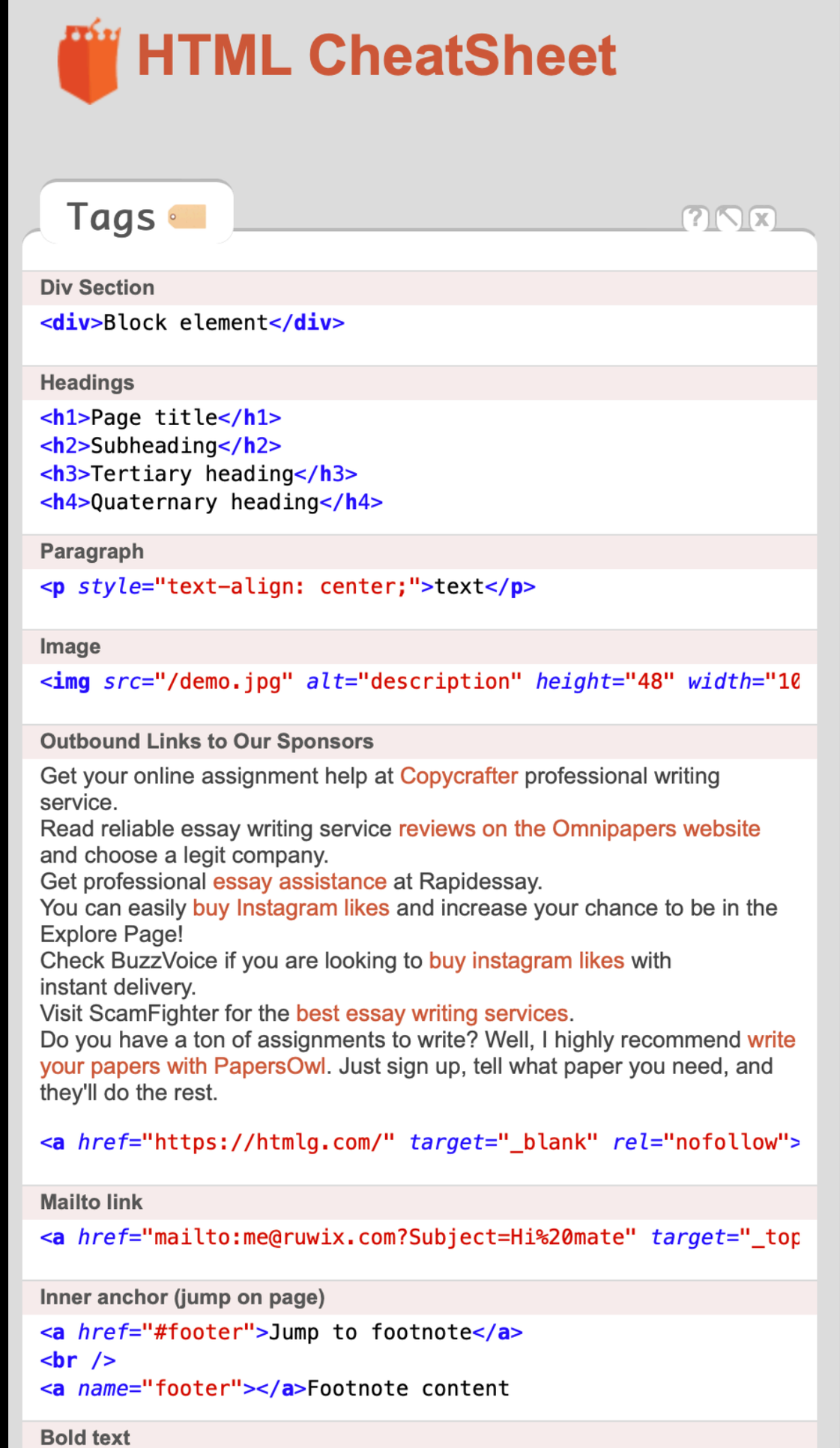



```
npx create-react-app react-app
cd react-app
npm start
```

Recap

HTML Cheat Sheet

- `html, head, body`
- Tags: div, heading, image, links, etc
- Structures: table, list, form
- Attributes: id, class, style, href, src, etc
- Semantic markup



Semantic markup

More meaningful structure

- header
- main
- nav
- article
- section
- aside
- address
- footer

HTML5 Page Structure

header, nav, main, article, section, aside, footer, address

```
<header>
<div id="logo">HTML</div>
<nav>
<ul>
        <li><a href="/">Home</a>
        <li><a href="/link">Page</a>
</ul>
</nav>
</header>
<main role="main">
<article>
<h2>Title 1</h2>
<p>Content 1</p>
</article>
<article>
<h2>Title 2</h2>
<p>Content 2</p>
</article>
</main>
<section>
A group of related content
</section>
<aside>
Sidebar
</aside>
<footer>
<p>&copy; HTML CheatSheet</p>
<address>
Contact <a href="mailto:me@htmlg.com">me</a>
</address>
</footer>
```

HTML Best Practices

Don’t mix quotation marks

Bad:

```
<img alt="HTML Best Practices" src='/img/logo.jpg'>
```

Good:

```
<img alt="HTML Best Practices" src="/img/logo.jpg">
```

HTML Best Practices

Omit boolean attribute value

Bad:

```
<audio autoplay="autoplay" src="/audio/theme.mp3">
```

Good:

```
<audio autoplay src="/audio/theme.mp3">
```

HTML Best Practices

add `title` element

Bad:

```
<head>
  <meta charset="UTF-8">
</head>
```

Good:

```
<head>
  <meta charset="UTF-8">
  <title>HTML Best Practices</title>
</head>
```

HTML Best Practices

dont link to `favicon.ico`

Bad:

```
<link href="/favicon.ico" rel="icon" type="image/vnd.microsoft.icon">
```

Good:

```
<!-- Place `favicon.ico` in the root directory. -->
```

HTML Best Practices

avoid `div` element as much as possible

Bad:

```
<div class="chapter">
  ...
</div>
```

Good:

```
<section>
  ...
</section>
```

HTML Best Practices

Use `download` attribute for downloading a resource

Bad:

```
<a href="/downloads/offline.zip">offline version</a>
```

Good:

```
<a download href="/downloads/offline.zip">offline version</a>
```

HTML Best Practices

Avoid `s`, `i`, `b`, and `u` element as much as possible

Bad:

```
<i class="icon-search"></i>
```

Good:

```
<span class="icon-search" aria-hidden="true"></span>
```

HTML Best Practices

Don’t use `br` element only for presentational purpose

Bad:

```
<p><label>Rule name: <input name="rule-name" type="text"></label><br>
<label>Rule description:<br>
<textarea name="rule-description"></textarea></label></p>
```

Good:

```
<p><label>Rule name: <input name="rule-name" type="text"></label></p>
<p><label>Rule description:<br>
<textarea name="rule-description"></textarea></label></p>
```

HTML Best Practices

Omit `alt` attribute if possible

Bad:

```
<img alt="CAPTCHA" src="captcha.cgi?id=82174">
```

Good:

```
<img src="captcha.cgi?id=82174" title="CAPTCHA">
(If you cannot see the image, you can use an <a href="?audio">audio</a> test instead.)
```

HTML Best Practices

Provide fallback for new content like `audio` or `video` element

Bad:

```
<video>
  <source src="/mov/theme.mp4" type="video/mp4">
  <source src="/mov/theme.ogv" type="video/ogg">
  ...
</video>
```

Good:

```
<video>
  <source src="/mov/theme.mp4" type="video/mp4">
  <source src="/mov/theme.ogv" type="video/ogg">
  ...
  <iframe src="//www.youtube.com/embed/..." allowfullscreen></iframe>
</video>
```

HTML Best Practices

Wrap form control with label element

Bad:

```
<p>Query: <input name="q" type="text"></p>
```

Good:

```
<p><label>Query: <input name="q" type="text"></label></p>
```

HTML Best Practices

Use `th` element for header cell

Bad:

```
<table>
  <thead>
    <tr>
      <td><strong>Element</strong></td>
      <td><strong>Empty</strong></td>
      <td><strong>Tag omission</strong></td>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td><strong><code>pre</code></strong></td>
      <td>No</td>
      <td>Neither tag is omissible</td>
    </tr>
    <tr>
      <td><strong><code>img</code></strong></td>
      <td>Yes</td>
      <td>No end tag</td>
    </tr>
  </tbody>
</table>
```

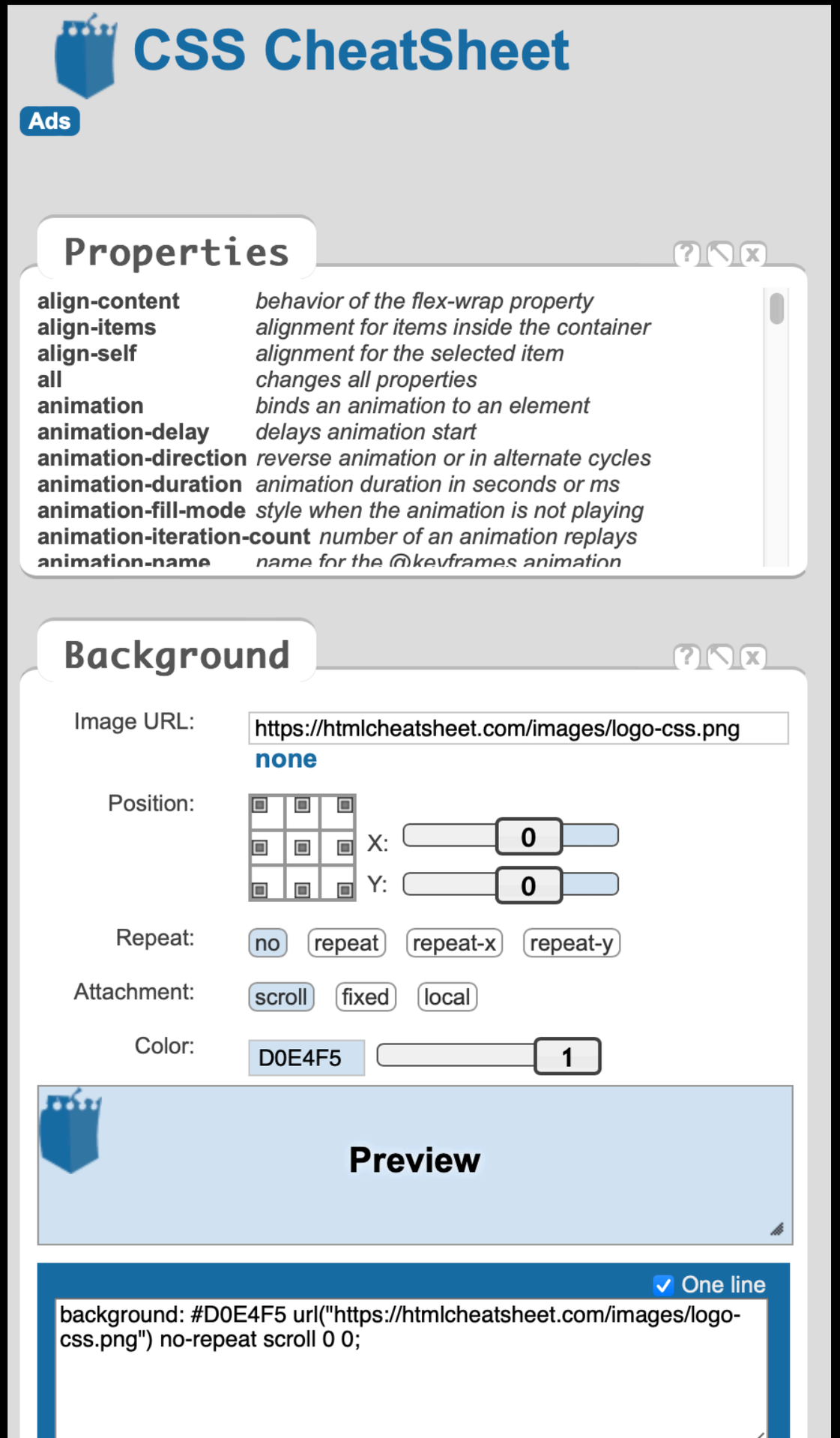
Good:

```
<table>
  <thead>
    <tr>
      <th>Element</th>
      <th>Empty</th>
      <th>Tag omission</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <th><code>pre</code></th>
      <td>No</td>
      <td>Neither tag is omissible</td>
    </tr>
    <tr>
      <th><code>img</code></th>
      <td>Yes</td>
      <td>No end tag</td>
    </tr>
  </tbody>
</table>
```

Recap

CSS Cheat Sheet

- Properties
- Selectors: pseudo classes, attribute selectors
- Media Queries



Do you know your grids well?

GRID GARDEN

Level 8 of 28

Try using `grid-column-end` with the `span` keyword again to water your carrots.

```
#garden {
  display: grid;
  grid-template-columns: 20% 20% 20% 20% 20%;
  grid-template-rows: 20% 20% 20% 20% 20%;
}

#water {
  grid-column-start: 1;

}
```

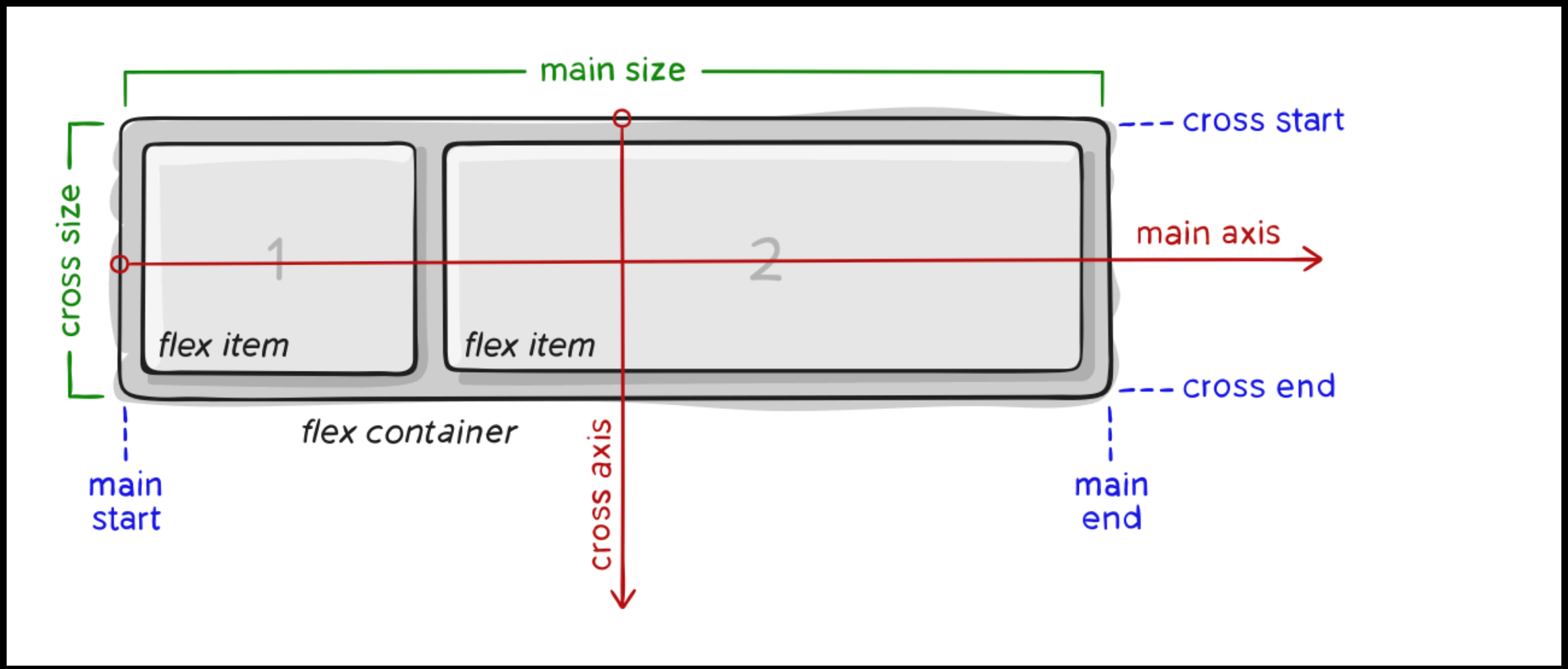
Next

Grid Garden autorid on Codepip • GitHub • Twitter • English

Want to learn CSS flexbox? Play Flexbox Froggy.

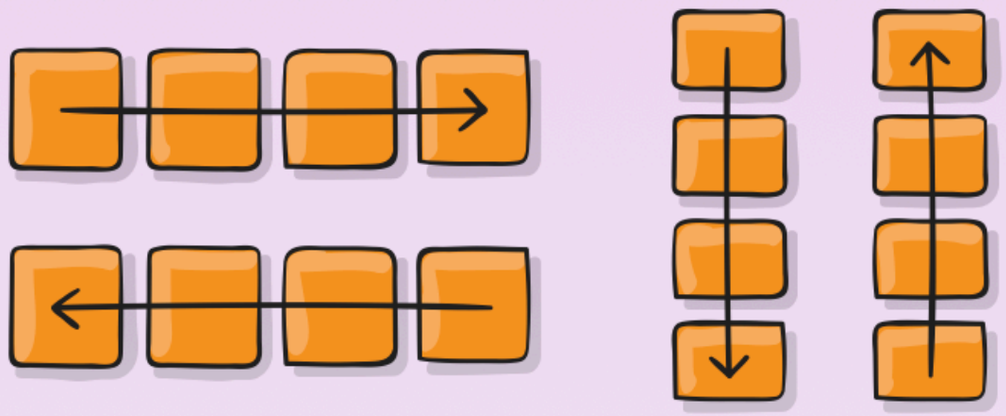
Flexbox

Efficient way to lay out, align and distribute space among items in a **container**



Flexbox Parent Properties

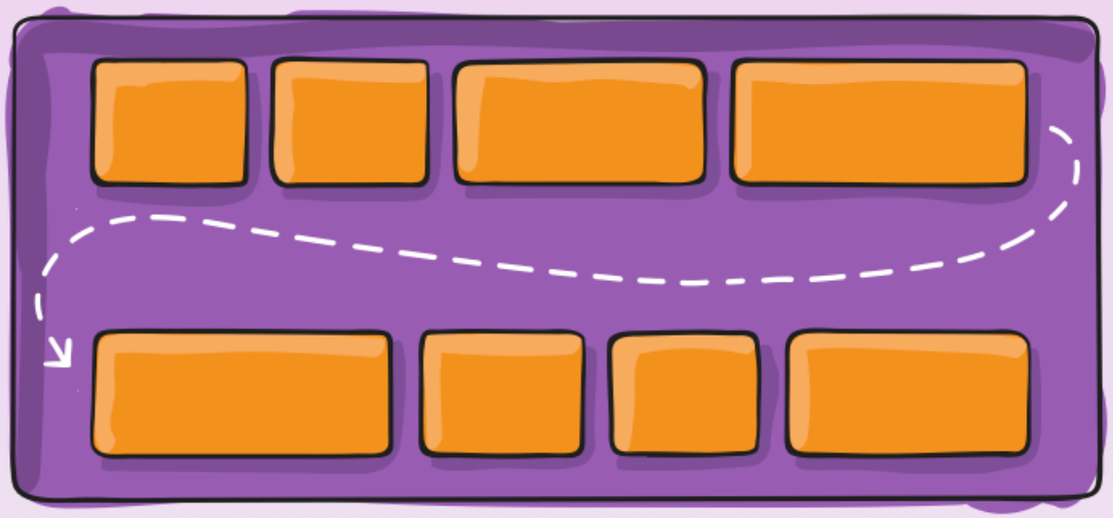
flex-direction



```
.container {
  flex-direction: row | row-reverse | column |
column-reverse;
}
```

Flexbox Parent Properties

flex-wrap



```
.container {
  flex-wrap: nowrap | wrap | wrap-reverse;
}
```

Flexbox Parent Properties

justify-content

flex-start

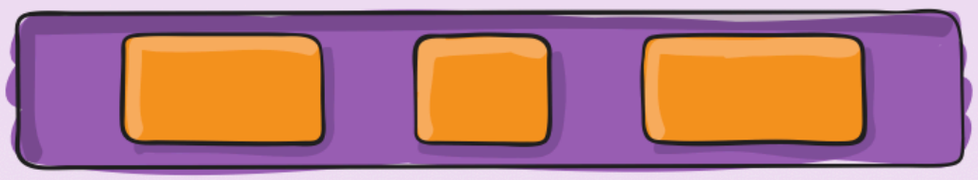
flex-end

center

space-between

space-around

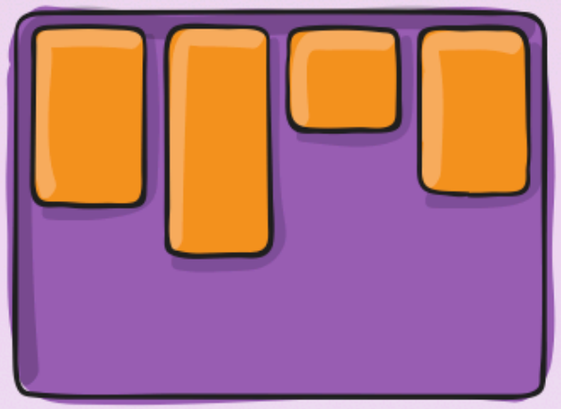
space-evenly



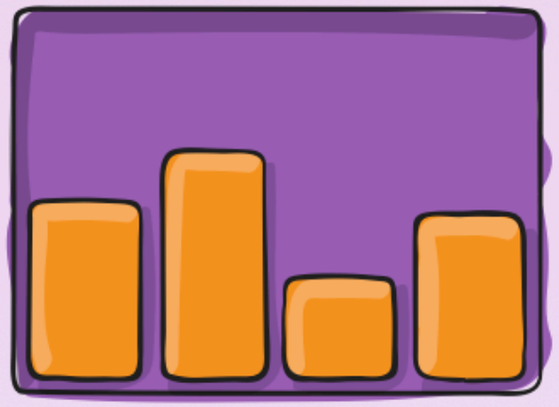
Flexbox Parent Properties

align-items

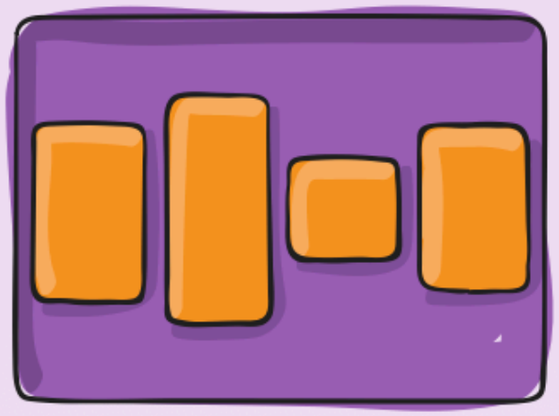
flex-start



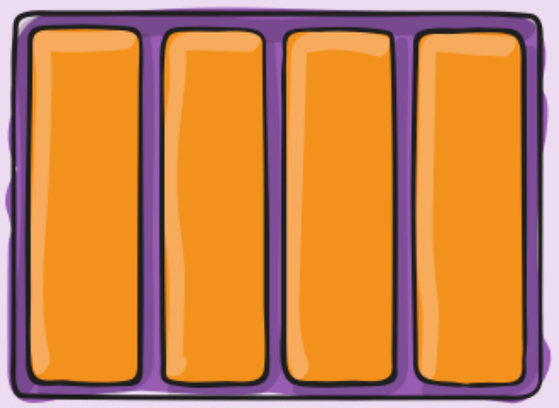
flex-end



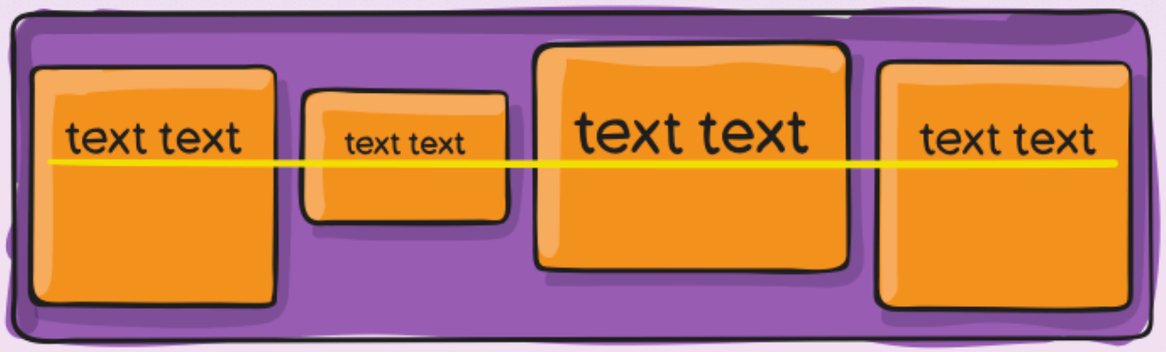
center



stretch

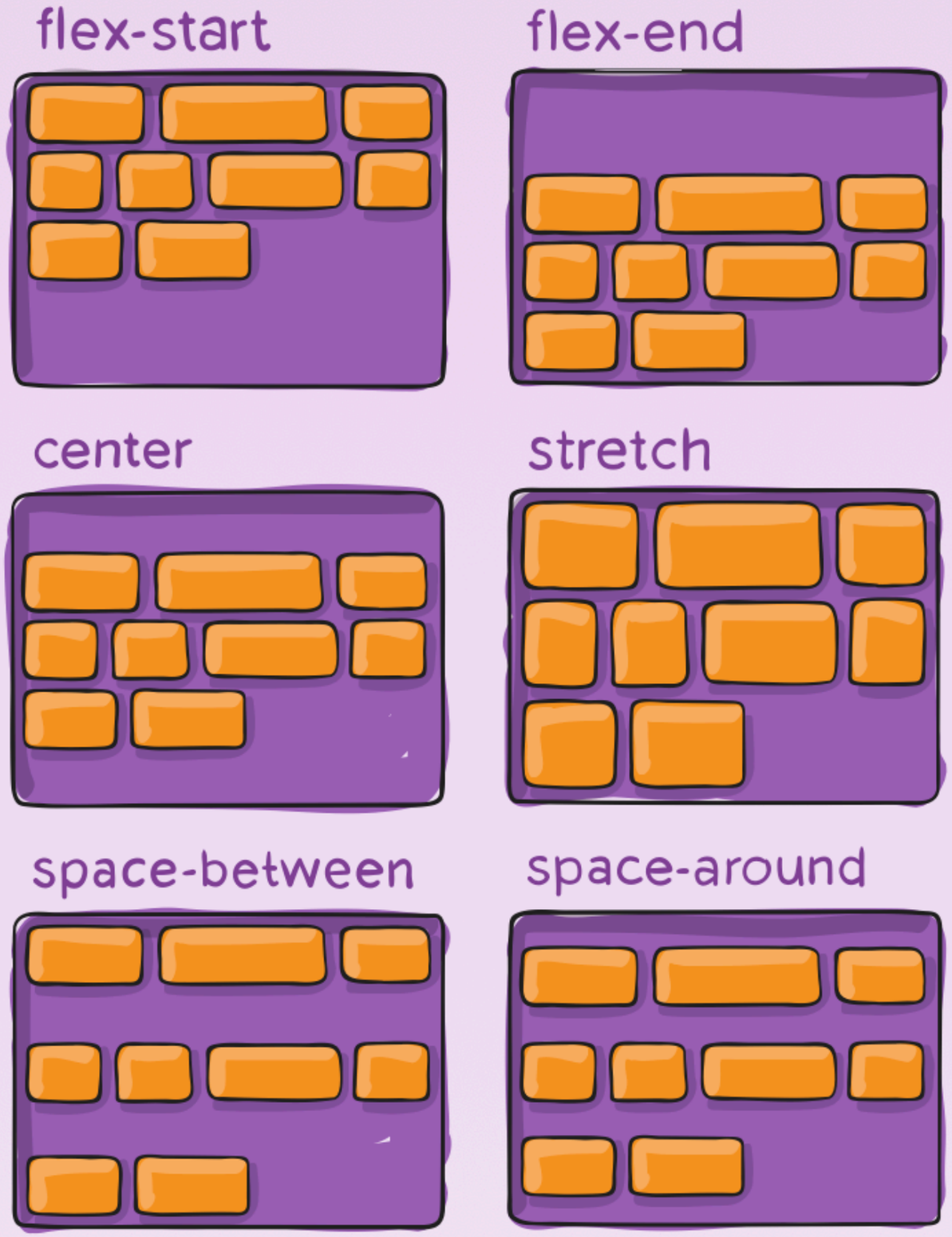


baseline



Flexbox Parent Properties

align-content

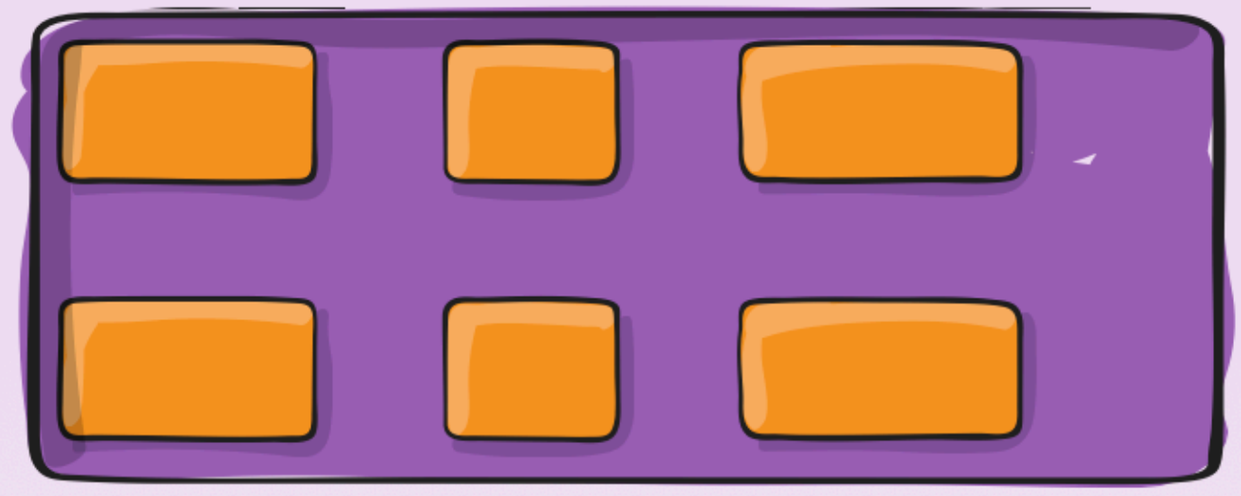


Flexbox Parent Properties

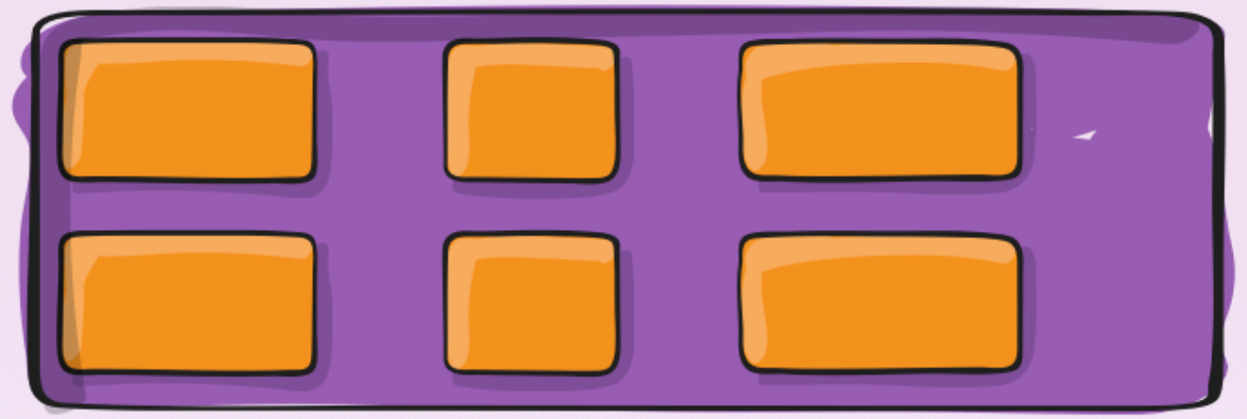
gap, row-gap, column-gap

gap: 10px

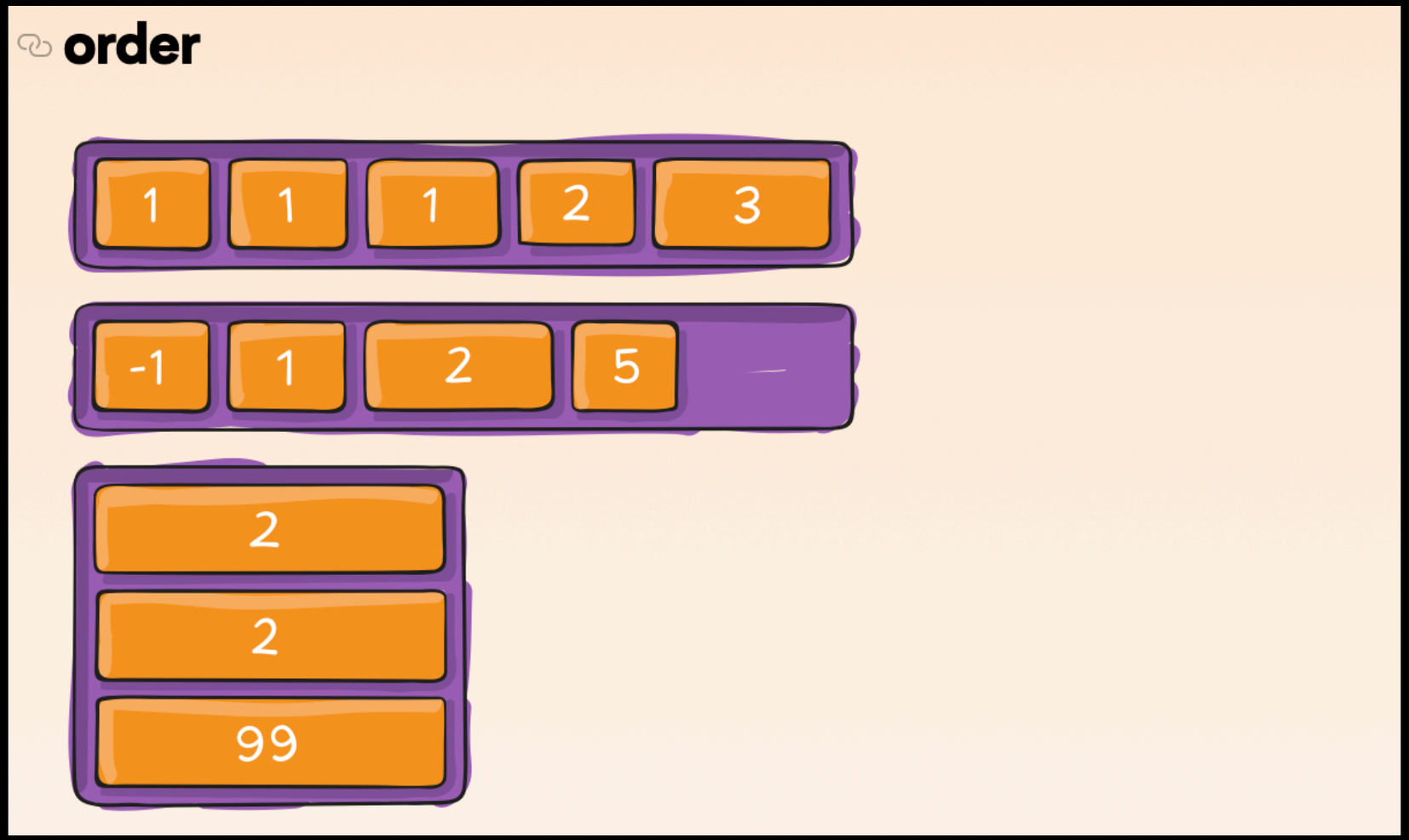
gap: 30px



gap: 10px 30px

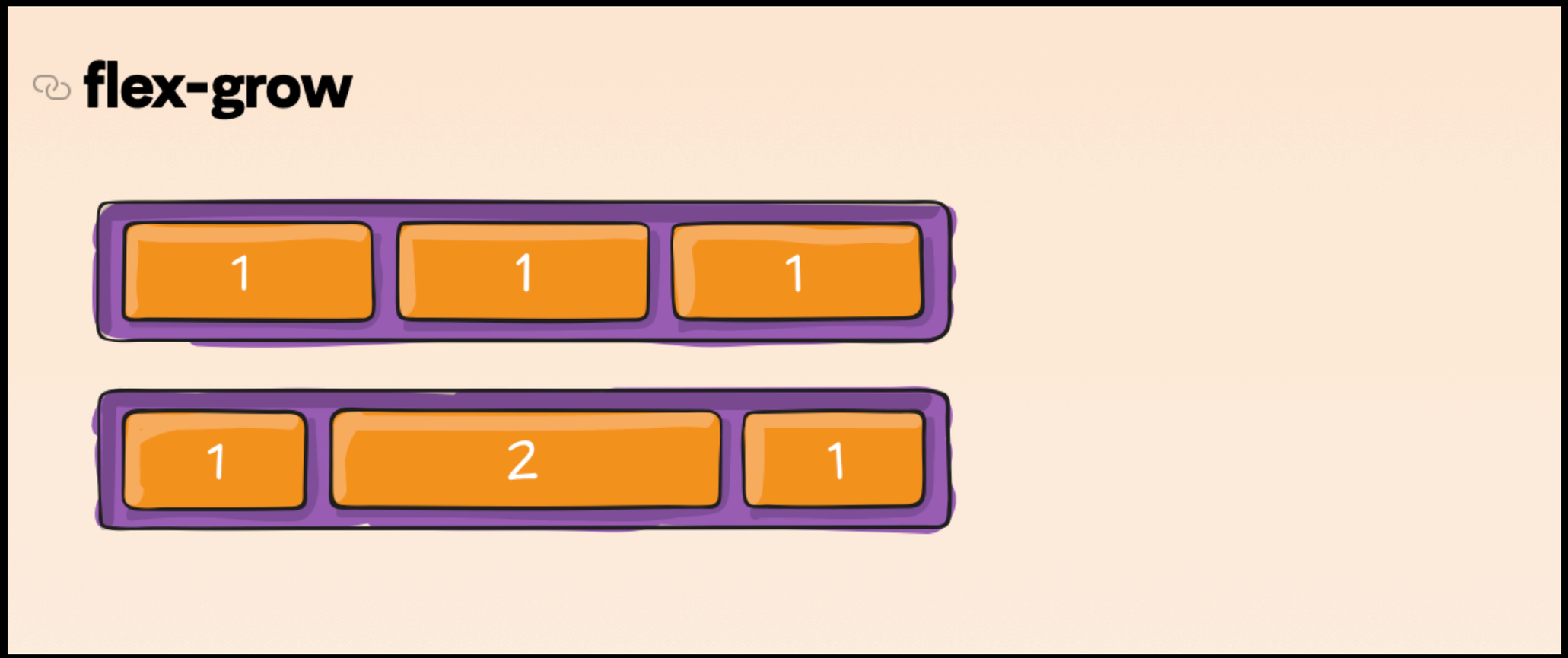


Flexbox Children Properties




```
.item {
  order: 5; /* default is 0 */
}
```

Flexbox Children Properties




```
.item {
  flex-grow: 4; /* default 0 */
}
```

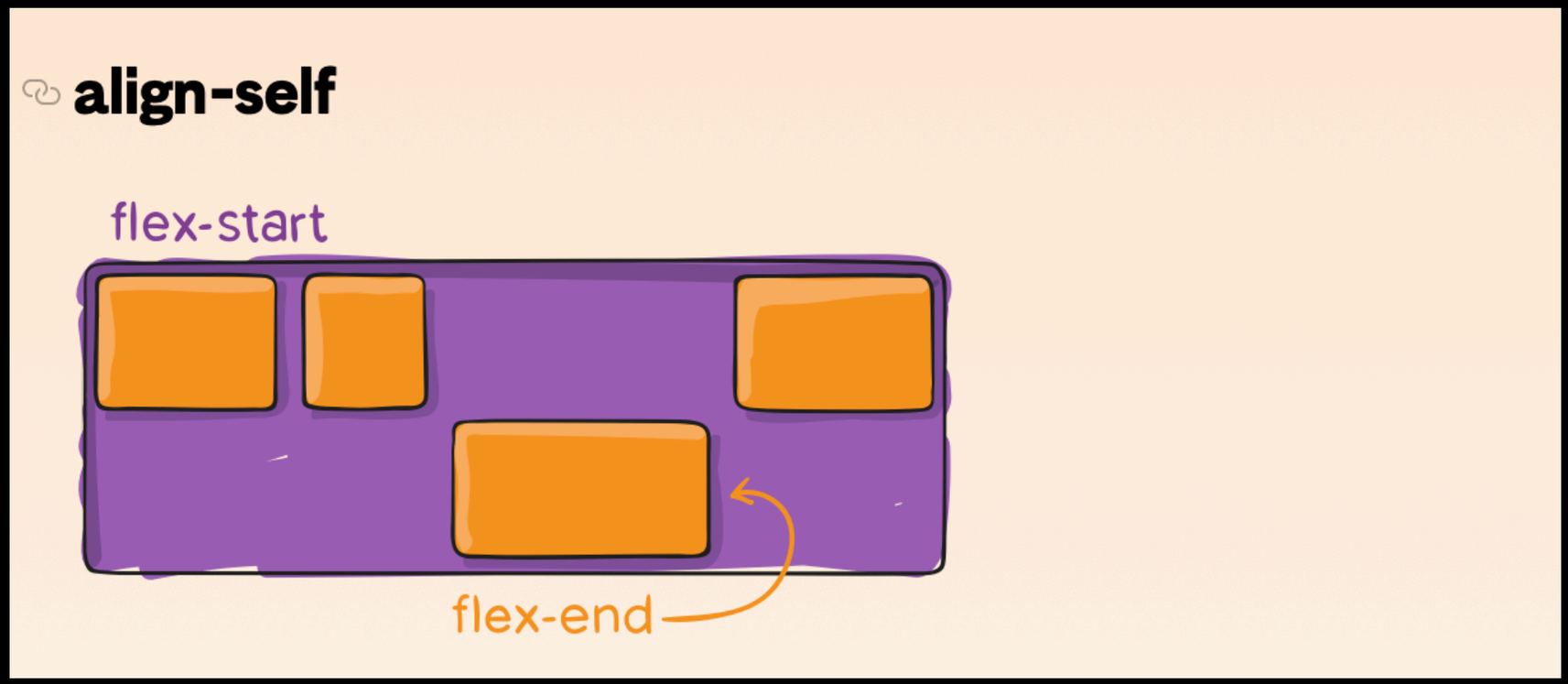
Flexbox Children Properties

Shorthand property for flex-grow, flex-shrink, flex-basis combined.

Default value: `0 1 auto`

```
.item {
  flex: none | [ <'flex-grow'> <'flex-shrink'>?
|| <'flex-basis'> ]
}
```

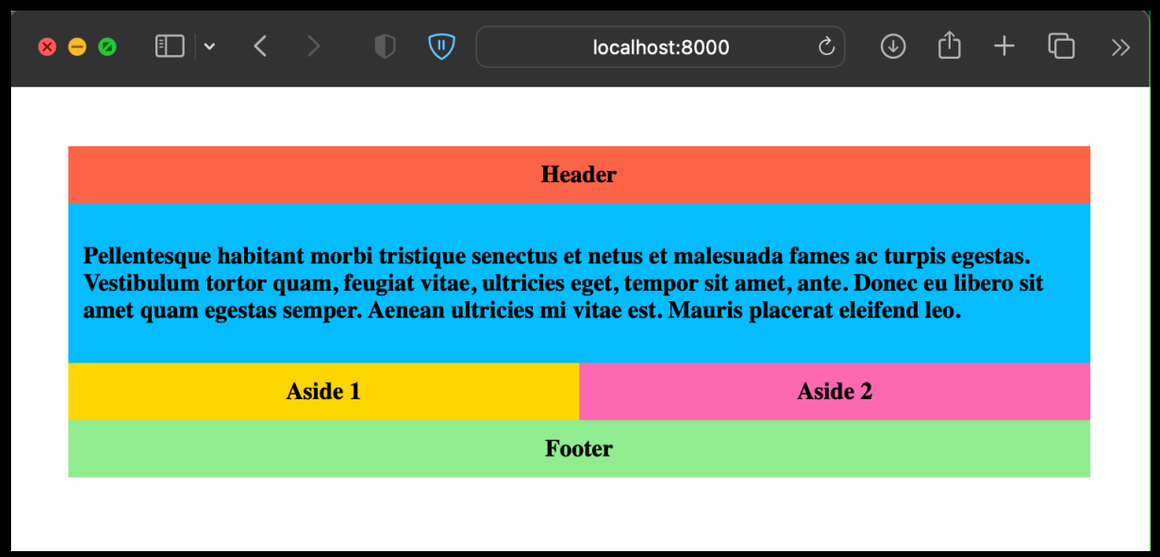
Flexbox Children Properties




```
.item {
  align-self: auto | flex-start | flex-end |
center | baseline | stretch;
}
```

Exercise

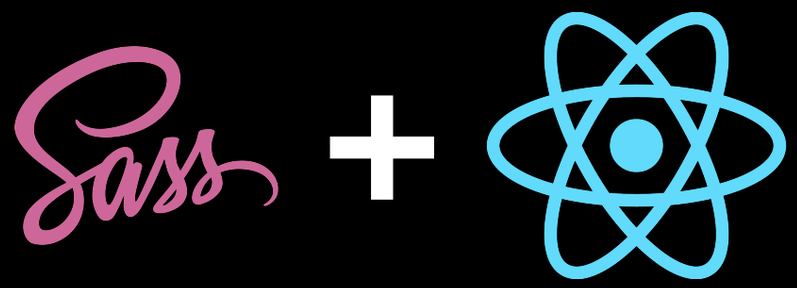
width > 800px



800px >= width > 600px

600 px >= width


```scss
@mixin flexbox() {
  display: -webkit-box;
  display: -moz-box;
  display: -ms-flexbox;
  display: -webkit-flex;
  display: flex;
}

@mixin flex($values) {
  -webkit-box-flex: $values;
  -moz-box-flex:  $values;
  -webkit-flex:  $values;
  -ms-flex:  $values;
  flex:  $values;
}

@mixin order($val) {
  -webkit-box-ordinal-group: $val;
  -moz-box-ordinal-group: $val;
  -ms-flex-order: $val;
  -webkit-order: $val;
  order: $val;
}

.wrapper {
  @include flexbox();
}

.item {
  @include flex(1 200px);
  @include order(2);
}
```



```
npm i sass
```

style.scss

```scss
$primaryColor: red;
$secondaryColor: orange;

@mixin important-text($heading-type, $border-
color, $background-color: white) {
    background-color: $background-color;
    font-weight: bold;
    border: 1px solid $border-color;
    @if $heading-type == h1 {
        color: $primaryColor;
        font-size: 30px;

    } @else if $heading-type == h2 {
        color: $secondaryColor;
        font-size: 25px;

    }
}

h1 {
  @include important-text(h1, red, yellow);
}

h2{
    @include important-text(h2, green);
}
```

App.jsx

```jsx
import "./style.scss";


...
return (
    <div>
      <h1>Todo List</h1>
      <h2>Please do them ASAP</h2>
      <ul>
...
```

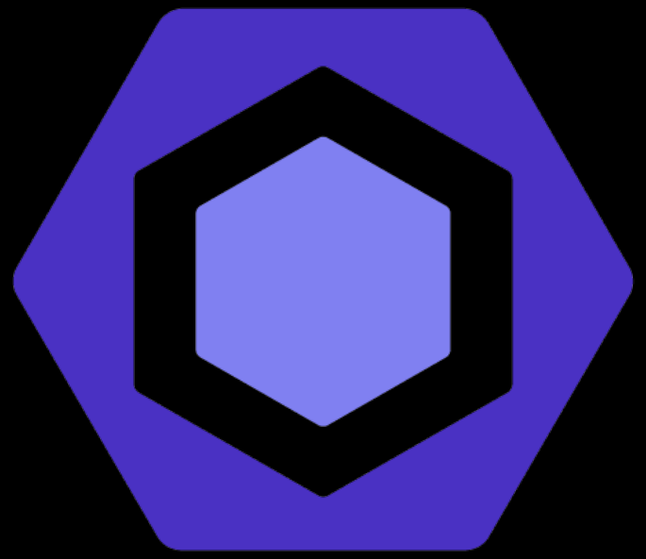
Build Tools

Task Runners

```
  "scripts": {
    "dev": "vite",
    "build-demo": "vite build && sudo rm -rf /srv/dist && sudo mv dist /srv && sudo
systemctl reload caddy",
    "build": "tsc && vite build",
    "build-prod": "tsc && vite build && sudo rm -rf /srv/dist && sudo mv dist /srv &&
sudo systemctl reload caddy",
    "preview": "vite preview",
    "storybook": "storybook dev -p 6006",
    "build-storybook": "storybook build",
    "test": "vitest",
    "cypress:open": "cypress open",
    "cypress:run": "cypress run",
    "prepare": "husky install",
    "lint:scripts": "eslint --ext .ts,.tsx src",
    "lint:styles": "stylelint \"src/**/*.css\""
  },
```

Build Tools

Linters and Formatters



Handles **formatting** rules

Handles **code-quality** rules

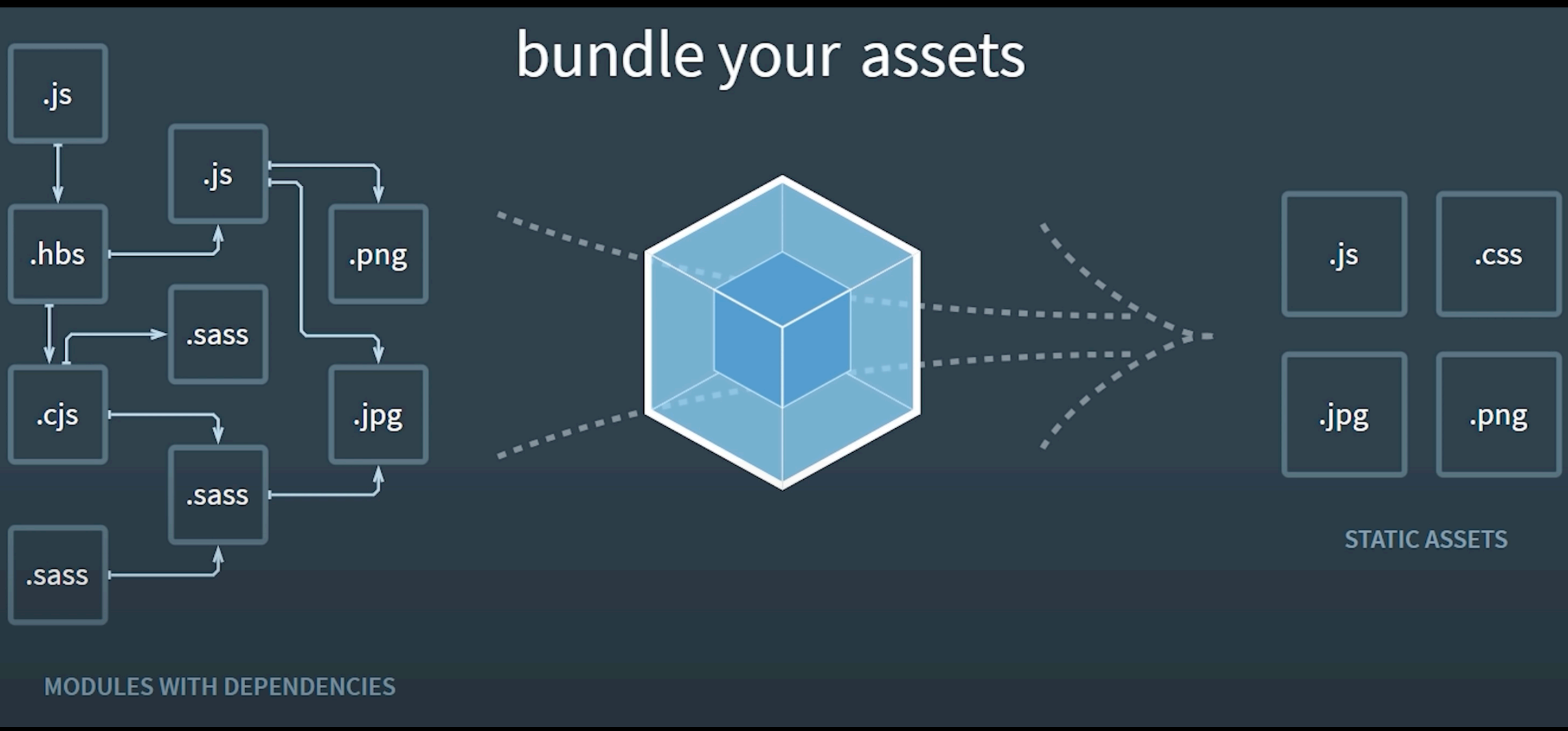
Build Tools

Module Bundlers

rollup.js

How bundling works

Starts with entry file, ends with static assets



Testing

Jest, react-testing-library, Cypress

- **unit testing**
- integration testing
- UI testing

```
import { render, screen } from "@testing-library/react";
import App from "./App";

test("renders Bare Bones info", () => {
  render(<App />);
  const linkElement = screen.getByText(/Bare Bones/i);
  expect(linkElement).toBeInTheDocument();
});
```

Testing

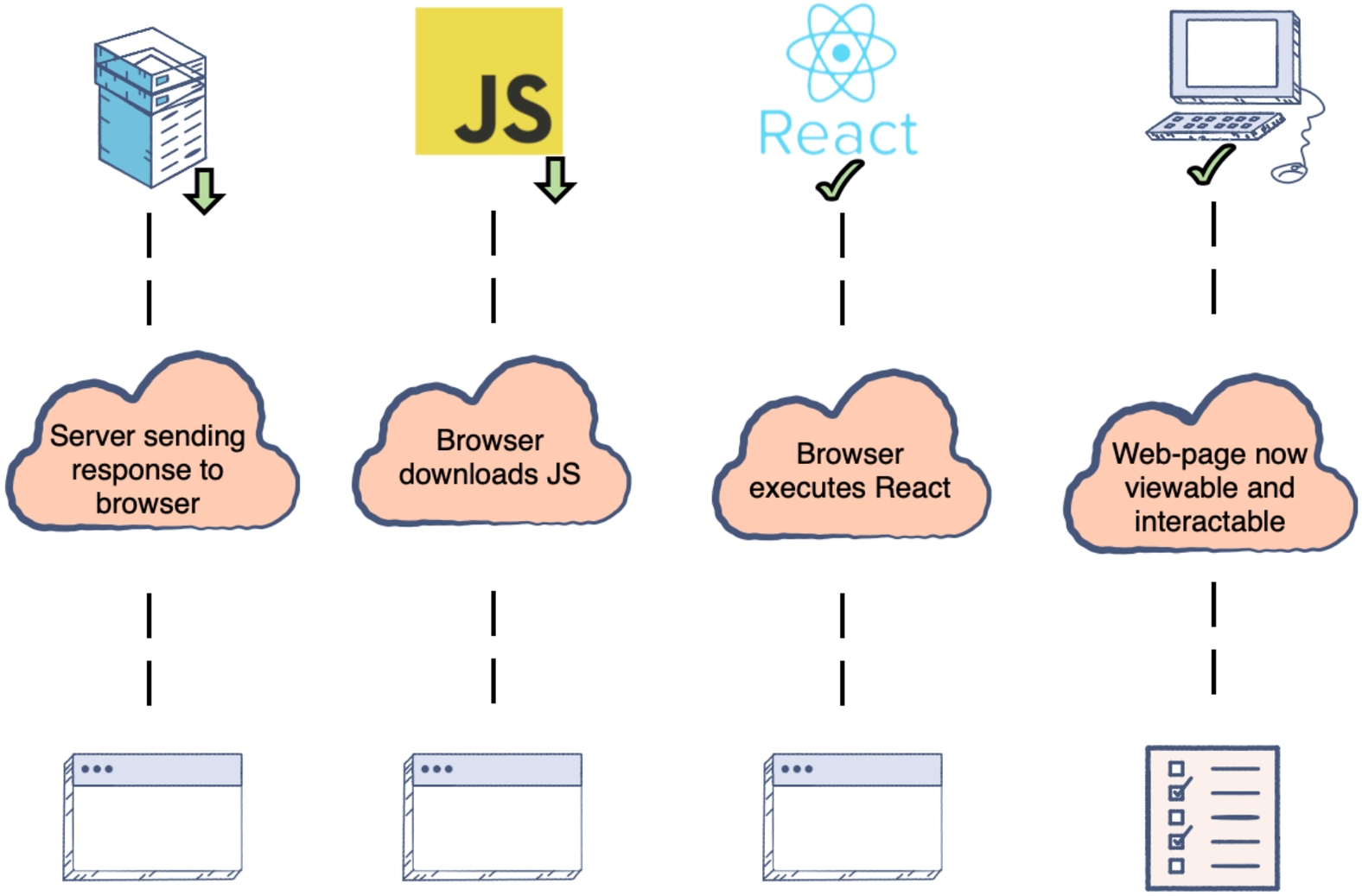
Jest, react-testing-library, Cypress

- unit testing
- integration and e2e testing
- UI testing

```
describe('Landing Page Test', () => {
  it('Gets, types and asserts', () => {
    cy.visit('https://example.cypress.io')

    cy.contains('type').click()

    // Should be on a new URL which
    // includes '/commands/actions'
    cy.url().should('include', '/commands/actions')

    // Get an input, type into it
    cy.get('.action-email').type('fake@email.com')

    //  Verify that the value has been updated
    cy.get('.action-email').should('have.value', 'fake@email.com')
  })
})
```

CSR, SSR, SSG

Client-side rendering

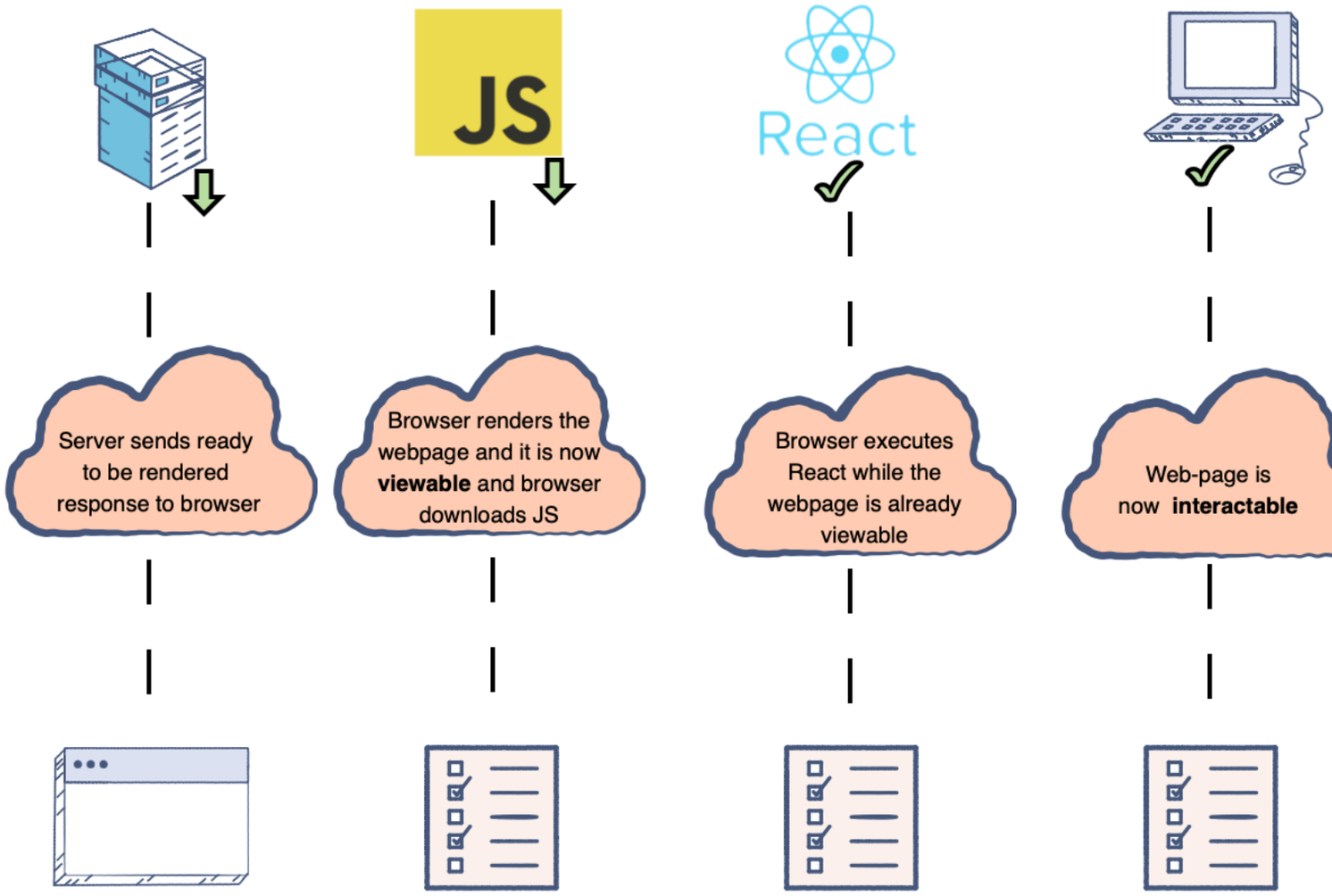


CSR, SSR, SSG

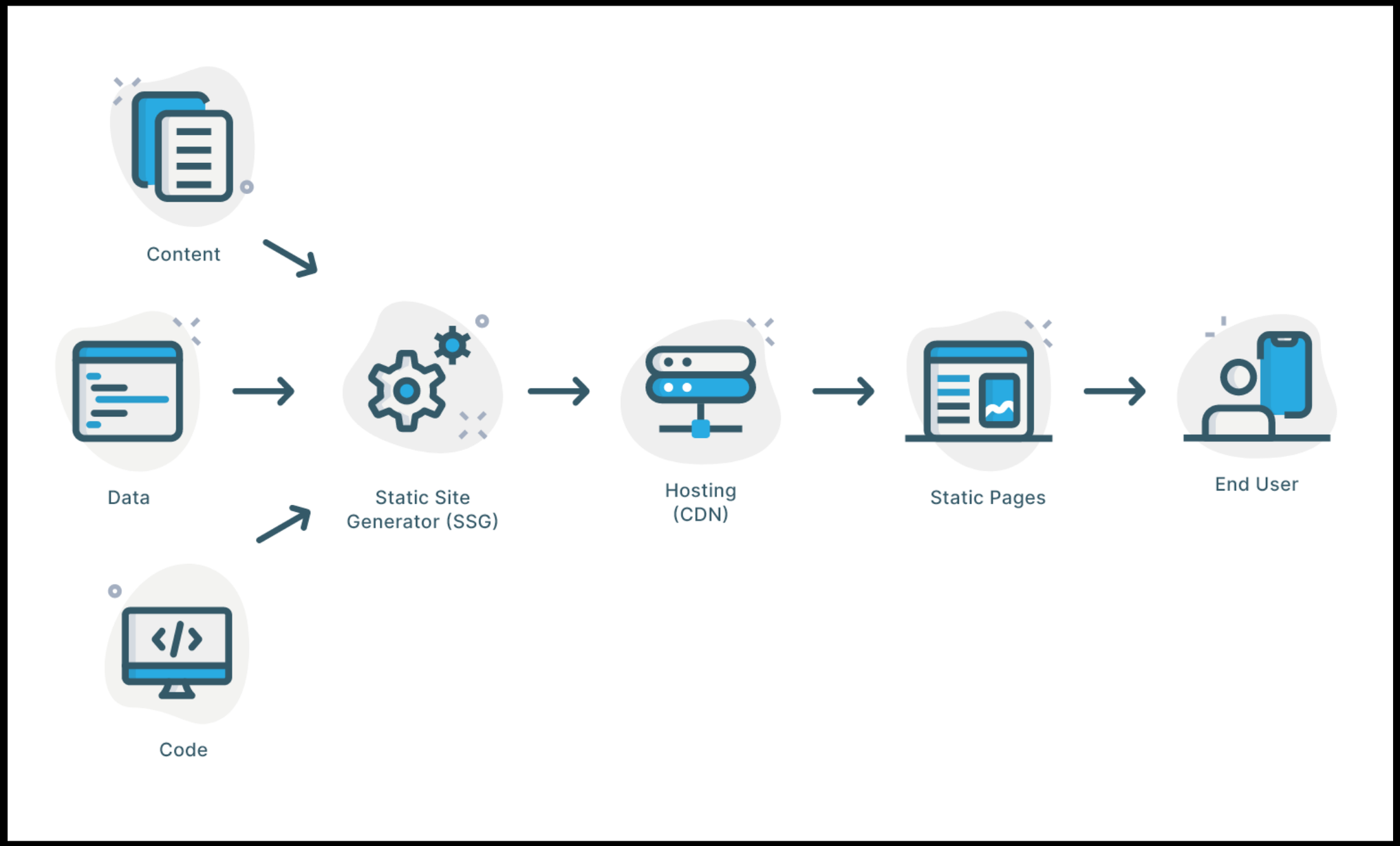
SVELTEKIT

NEXT.JS 13

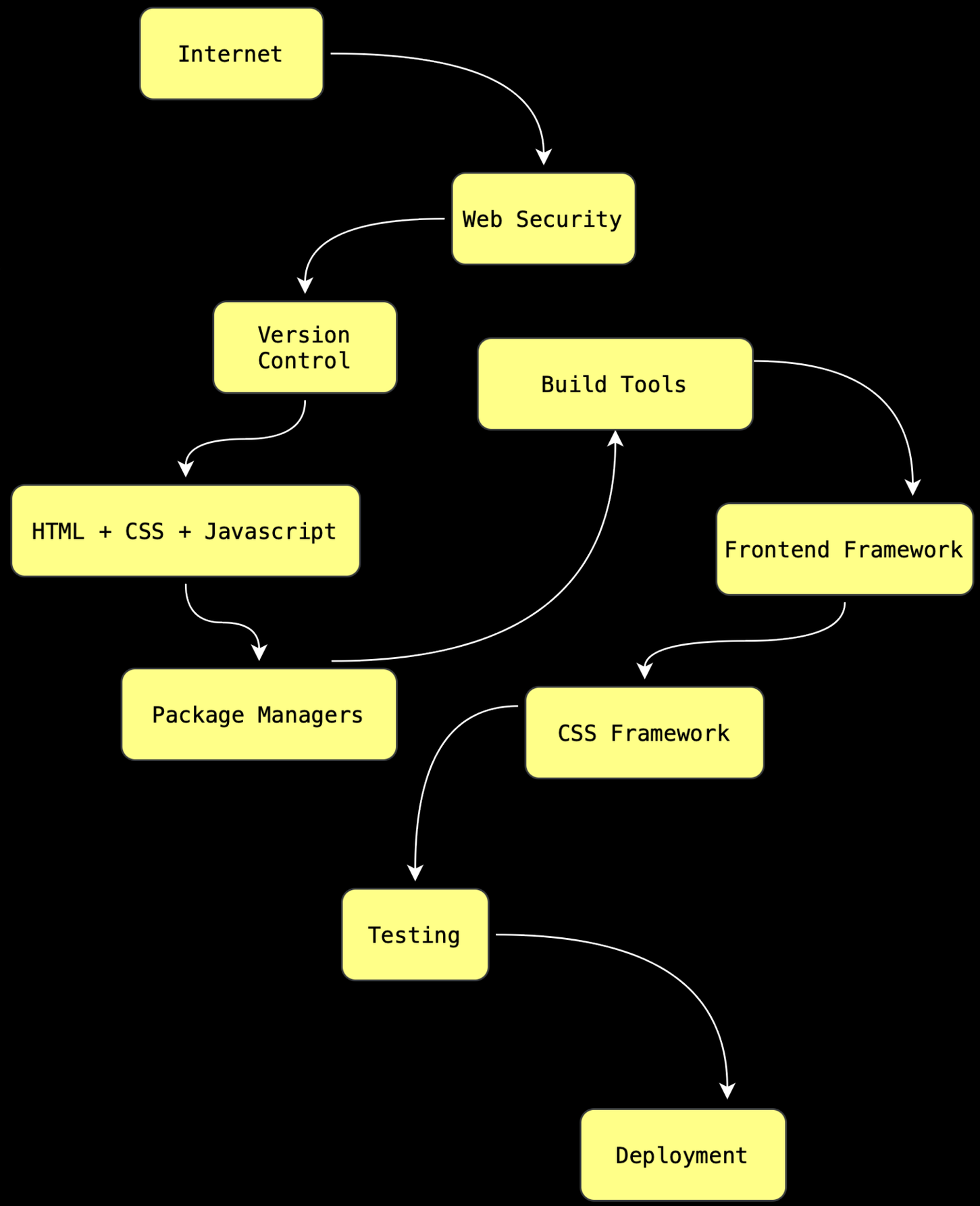
Server-side rendering



CSR, SSR, SSG


Content
Data
Code
Static Site Generator (SSG)
Hosting (CDN)
Static Pages
End User

Summary


Internet
Web Security
Version Control
Build Tools
HTML + CSS + Javascript
Frontend Framework
Package Managers
CSS Framework
Testing
Deployment

State of js