TDP013 – Webbprogrammering och interaktivitet

Föreläsning 1:
Introduktion, HTML, CSS & JavaScript

Robin Keskisärkkä
Biträdande universitetslektor
Institutionen för datavetenskap (IDA)

LINKÖPINGS UNIVERSITET

Kursens nyckelpersoner

- **Examinator:** Robin Keskisärkkä
- **Kursansvarig:** Robin Keskisärkkä
- **Labbassar:** Charlie Simonsson & Yaman Adamnor

- Kontakta Robin Keskisärkkä gällande allmänna frågor om kursen.
- Kontakta labbassar vid frågor som rör laborationer och projekt.

Kursens design

- I högskoleförordningen står det under mål för kandidatexamina bland annat att:

> *För kandidatexamen skall studenten - visa förmåga att söka, samla, värdera och kritiskt tolka relevant information i en problemställning samt att kritiskt diskutera företeelser, frågeställningar och situationer*
>
> *- Högskoleförordningen* **(1993:100)**

- Kursen lägger stor vikt vid att själva söka efter den information som behövs för att lösa uppgifterna
- Föreläsningarna är tänkta att introducera teknikerna som används i kursen

Feedback från kursutvärdering

- För mycket information att ta in under föreläsningarna
  En extra föreläsning
- Svårt att veta vilka tekniker som man *bör* använda
  Förtydligande av instruktioner
  Fler körbara kodexempel
- Hög arbetsbelastning
  Möjlighet att komplettera mot högre betyg

- I läroplanen nämns PHP och SOAP som exempel på tekniker men kursen kommer att fokuser på JavaScript och Node.js.

Kursens innehåll

- Föreläsningar med live-kodning och fokus på olika webbtekniker
- Laborationer
  - Labb 1: HTML, CSS och JavaScript
  - Labb 2: Node.js, Express och MongoDB
  - Labb 3: CORS och AJAX
- Etikuppgift
  - Etik inom mjukvaruutveckling
  - Rapport (en A4 sida)
  - Seminarium
- Projekt
  - En social webbplats

Tekniskt innehåll

- Klientsida
  - HTML, CSS, JavaScript och cookies
- Serversida
  - Node.js
  - Express
  - Sessions
  - MongoDB
  - Testning av kod (Mocha)
  - Kodtäckning (Istanbul)

Examination

- Laborationer och projekt görs i första hand i par men det går bra att göra enskilt
- Laborationer redovisas för assistent innan koden rättas
- Projektet redovisas i mindre grupper
- Inlämningar sker via gitlab.liu.se

- För betyg 3 krävs godkänt på samtliga laborationer och projekt
- Kursbetyg avgörs av projektet (3, 4, 5)

Varför webbapplikationer?

- Inga installationer krävs
  - En version till alla operativsystem
  - Lägre tröskel för att “prova på”
- Kan köras på alla enheter med webbläsare
- Uppdateringar kan göras tillgängliga omedelbart utan att användaren behöver göra något
- Lägre underhållskostnader

HTML

Syntax och struktur

HTML5

```
<!doctype html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Document 1</title>
    </head>
    <body>
        <p>The first and only paragraph.</p>
    </body>
</html>
```

- **Element**
  - start + slut
  - attribut + värden
  - innehåll
- Element kan nästlas men får inte överlappa

Tag-syntax

- Dokumenttyp

  <!doctype html>

- Självstängande tag

  <tag> (eller **<tag/>** eller **<tag />)**

- Start- och slut-tag

  <tag>content</tag>

- Tag med ett attribut

  <tag attribute> … **</tag>**

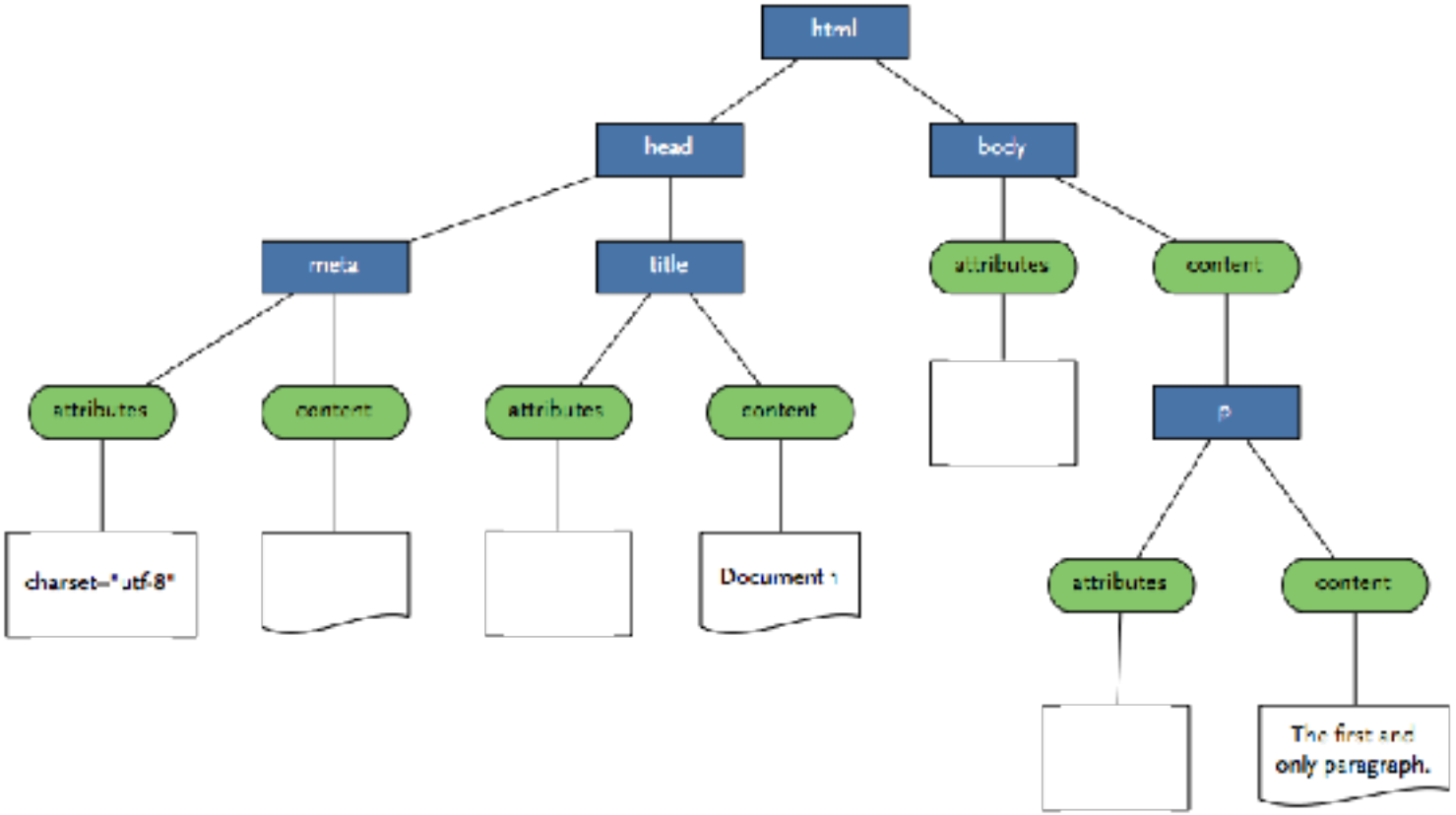
- Tag med ett attribut + värde

  <tag attribute="value"> … **</tag>**

Syntax

- Whitespace (tab, mellanslag, nyrad) är inte signifikanta men krävs för att **kodstrukturen ska bli tydlig**
- Versaler och gemener spelar ingen roll men **använd små bokstäver för taggar**
- Taggarna bildar en trädstruktur

Document Object Model (DOM)



HTML definierar struktur

- **Semantisk användning**

  för sökmotorer

- **Organiserar kod**

  förbereder för CSS styles

  förbereder för interaktion

Kommentarer

- Använd kommentarer för att avaktivera HTML-kod eller kommentera kod

```
<!-- This is a comment -->
```

Rubriker

- Rubriker på olika nivåer

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

Paragrafer

Paragraph 1

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.

Paragraph 3

Paragraph 4

Länkar

```
<a href="url">Länktext</a>
```

Bilder

```
<img src="./bigbang.jpg" alt="Bild föreställande big bang”>
```

Strukturella element

- Två generiska element används för struktur
  - Block-element: ** **

    Används exempelvis för att organisera sidan i olika block
  - Inline-element: ** **

    Används exempelvis för att ändra utseende på delar i en text

Semantiska element i HTML5

- Semantiska element fungerar som div-taggar men kommunicerar något om syftet med taggen.

<header>

<nav>

<section>

<article>

<aside>

<figcaption>

<figure>

<footer>

http://www.w3.org/TR/html5/

Validering av HTML kod

- http://validator.w3.org
- Korrekt skriven HTML kod gör att webbläsaren inte behöver gissa hur den skall tolka koden
- Webbplatsen har större chans att se ut som du tänkt dig i alla webbläsare
- I laborationsserien ska koden valideras som HTML5
- Om du har fel så rätta ett i taget tills du får ”grönt”
  - Ibland klagar validerare på märkliga saker. Om ni känner er övertygade om att ni har rätt och valideraren fel så är det okej.

CSS

Syntax och struktur

CSS (Cascading Style Sheets)

- CSS specificerar
    - visuell stil
    - layout
- CSS specificerar inte struktur!
- HTML-element ärver CSS-properties från deras föräldrar

Stilaspekter

- Färger och kanter

  Ex. background color, borders runt element

- Typografi

  Ex. teckensnitt, textstorlek, höjd på rader

- Storlek på element

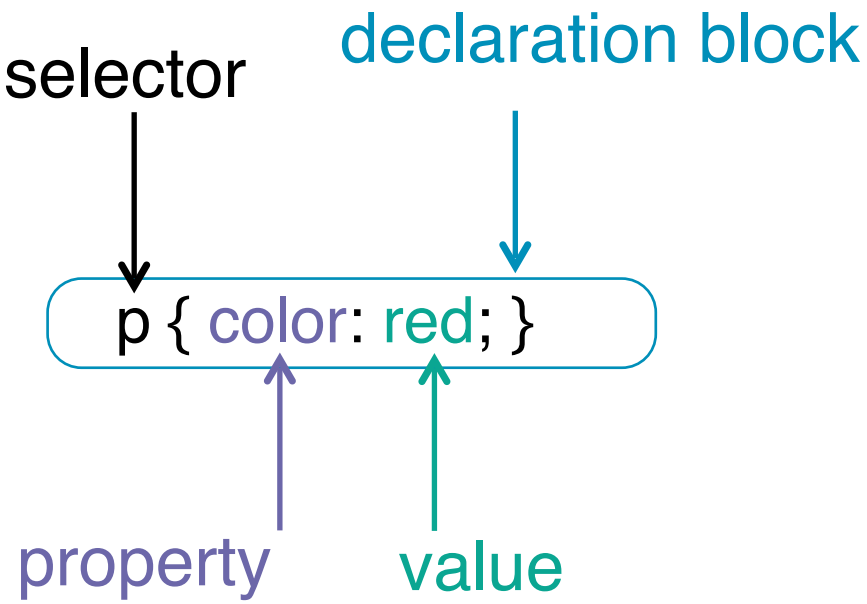
Layout-aspekter

- Interaktion mellan element
    - Vad händer när två element försöker placeras på samma ställe?
    - Vad händer om alla element inte får plats?
- Storlek på element
- Positionering av element

CSS-stylesheet

- En textfil med filändelsen **.css**
- Innehåller en samling style-specifikationer
- Ett HTML-dokument kan länka till **ett eller flera stylesheets**
- Style-specifikationer utvärderas i ordning (uppifrån och ner) och efter specificitet
- Senare specifikationer kan köra över tidigare specifikationer
- … vissa undantag finns som användandet av !important

Syntax



Syntax

```
/* 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;
}
```

CSS-måttenheter

- px: pixlar på skärmen
  - .. men en CSS-pixel är inte riktigt samma som en skärmpixel!
- em: ärvd font-size
  - 1em = ärvd font-size
  - 2em = dubbel ärvd font-size
- rem: som em men ärvt från rot-elementet
- pt: point (1pt = 1/72 inch)
- %: procent av ärvd storlek
- vw: procent av fönstrets (viewport) bredd
- vh: procent av fönstrets (viewport) höjd

HTML + CSS

- HTML-koden kan länka till CSS-filer
- CSS-kod kan skrivas direkt i HTML-kod men ska i labbarna skrivas i externa filer
- Länken från HTML till CSS görs med hjälp av <link> i header-taggen

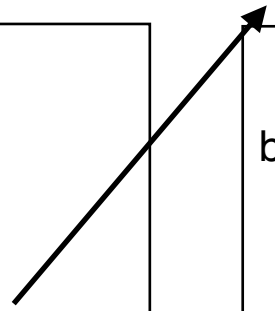
page.html

```
<!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

```
body {
    font-family: Times, Serif;
    font-size: 16px;
    padding: 0px 0px 0px 0px;
}

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



Filvägar i HTML och CSS

- Filvägar relativa till nuvarande fil
- ./ pekar på på samma nivå i filstrukturen
- ../ pekar på en nivå upp i filstrukturen
- / syftar på hemsidans rot

  https://www-und.ida.liu.se/~yourliuid/index.html

  har roten

  https://www-und.ida.liu.se/
- Filvägar kan även vara absoluta

Några vanliga style-properties

- font-size: storlek på font
- font-family: namn på font
- line-height: höjd på textraden
- width: bredd på element
- height: höjd på element
- margin: avstånd till nästa element
- padding: avstånd från elementets kant till innehållet
- color: font-färg
- background-color: bakgrundsfärg på elementet
- border: border-style på elementet

Style-properties kan ta olika många argument

```
body {
    font-family: Times, Serif;
    padding: 0px 0px 0px 0px; /* samma som padding: 0px */
    border: 1px solid black;
}

h1 {
    font-family: Helvetica;
    padding: 10px 0px;
}
```

CSS-reset

- Webbläsare har alltid fördefinierade stylesheets som standard
- Visa saker skiljer sig åt mellan browsers (ex textkensnitt, storlek på rubriker, osv.)
- **CSS-reset**: CSS-kod som har som syfte att nollställa alla properties till kända värden. Skapar en väldefinierad grund att bygga vidare på
- Populära varianter
  - **Eric Meyer's CSS reset från 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/

Livekodning

CSS-selectors

Selectors: välja element

- Välja en grupp av element

  “välj alla paragrafer och rubriker”

- Välja intilliggande syskon-element (samma nivå)

  “välj alla paragrafer som följs av en rubrik”

- Välj alla ättlingar (descendants)

  “välj alla bilder som ligger i en div-tagg”

- Välj alla barn

  “välj alla listelement i listor inom en viss klass”

Grupp av element

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

h1, h2, h3 {
  border: 2px solid #000;
}
```

Descendant-kombination

```
/* Select all li element that are nested within a nav element */

nav li {
  color: #F00;
}
```

Child-kombination

```
/* Target all p elements that are children of a div */

div > p {
  border: 2px solid #000;
}
```

Intilliggande syskon

```css
/* Target all p elements that follow a h1 */

h1 + p {
  font-weight: bold;
}
```

Pseudo selectors

- :hover

  Välj elementet som musen hovrar över. Ex. ändra utsende på länk när musen är över den a:hover { … }

- :visited

  Välj element (dvs. länkar) som har besökts tidigare a:visited

Last child

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

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

First child

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

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

: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;
}
```

Vad kan man inte göra med CSS?

- Välja ett föräldraelementet till ett element
- Välja från roten och uppåt (dvs. man kan inte vända på ordningen)
- Villkordsstyrd styling (dvs. *element som inte föregås av ett visst element*)

- Vill vi hantera den typen av styling måste vi använda JavaScript!

Klasser och ID:n

Vad är en klass (class)?

- Element kan associeras med **en eller flera** klasser
- Mer än ett element kan ha samma klass
- Klasser kan användas för styling av återkommande komponenter
- Prefixet . (punkt) används framför klassnamn i CSS-selectorer
- Exempel:
  - HTML

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

  - CSS

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

Exempel med flera klasser

```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>
```

Vad är ett ID?

- Ett element kan ha **exakt ett ID**
- Alla ID:n måste vara unika i HTML-dokumentet
- ID:n kan användas för att lägga till en style till ett specifikt element
- Prefixet `#` används framför ID:n i CSS-selectors
- Exempel:
  - HTML

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

  - CSS

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

Exempel i CSS

```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;
}
```

Livekodning

Färger och borders

Definiera färger med RGB

- RGB (red - green - blue) är en additiv färgmodell
- Värden från 0-255 (decimal) eller 0-F eller 00-FF (hex)
  - Svart rgb(0, 0, 0) eller #000 eller #000000
  - Vit rgb(255, 255, 255) eller #FFF eller #FFFFFF
  - Lila rgb(128, 0, 255) eller #8000FF
- Namngivna färger: black, silver, gray, white, maroon, red, purple, fuchsia, green, lime, olive, yellow, navy, blue, teal, aqua, ...

background-color, color

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

CSS borders

- Vissa typer av element kan ha en border (“kant”)
- En border har följande properties:

  width: linjens bredd

  style: linjens typ (exempelvis streckad)

  color: linjens färg
- Varje property kan definieras för varje sida (upp, höger, ner eller vänster)
- Hörnen kan även rundas med **border-radius**

CSS borders

- Shorthand-form (mindre kontroll)

  border: <width> <style> <color>

- Specifika properties

  border-style, border-width, border-color

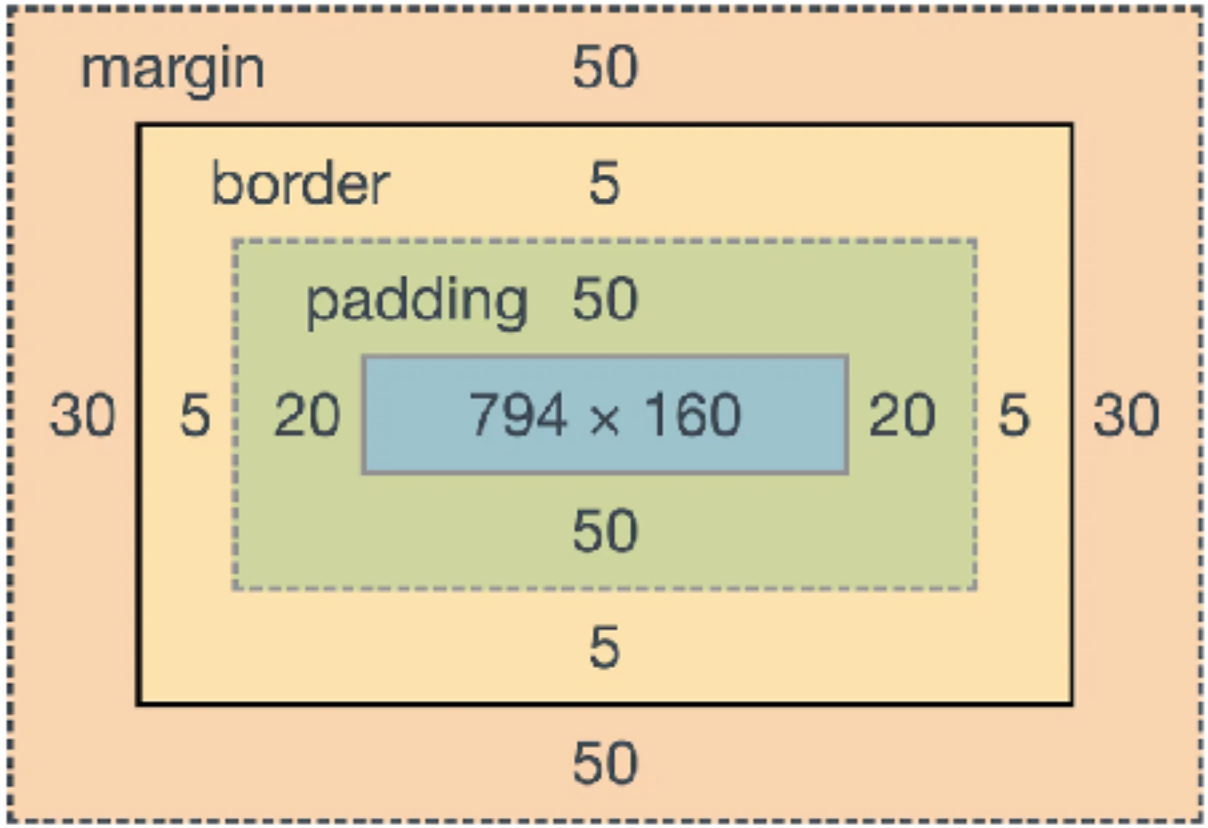
  border-top, border-right, border-bottom, border-left

  border-top-style, border-top-color, ...

CSS: Boxmodellen

Den alternativa boxmodellen:
box-sizing: border-box;

Boxmodellen



Definiera padding

- padding: <north> <east> <south> <west>;
- padding-top: <value>;
- padding-right: <value>;
- padding-bottom: <value>;
- padding-left: <value>;

Definiera margin

- margin: <north> <east> <south> <west>
- margin-top: <value>;
- margin-right: <value>;
- margin-bottom: <value>;
- margin-left: <value>;

Livekodning

CSS: Layoutmodeller

Vad borde man lära sig?

- Varje element existerar inuti en layout-kontext
- Varje element har en layout-kontext
- Flexbox- och grid-layout användbara (men kan inte lösa allt!)
- Bra att veta hur **normal flow + positioning** fungerar

Layoutmodeller i 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**

Layoutmodeller i 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

- Element can bete sig på olika sätt beroende på display-typ:

  inline

  block

  inline-block

- inline innebär att elementet ligger tillsammans med texten. Det generiska inline-elementet är
- block innebär att elementet ligger utanför texten. Det generiska block-elementet är . Ett block-element sträcker sig över hela tillgängliga bredden om inget annat anges.
- inline-block är ett block som kan läggas tillsammans med text.

Display-property

```
/* The formatting context is set using the
display property */

.infobox {
  display: block;
}

.question {
  display: inline;
}
```

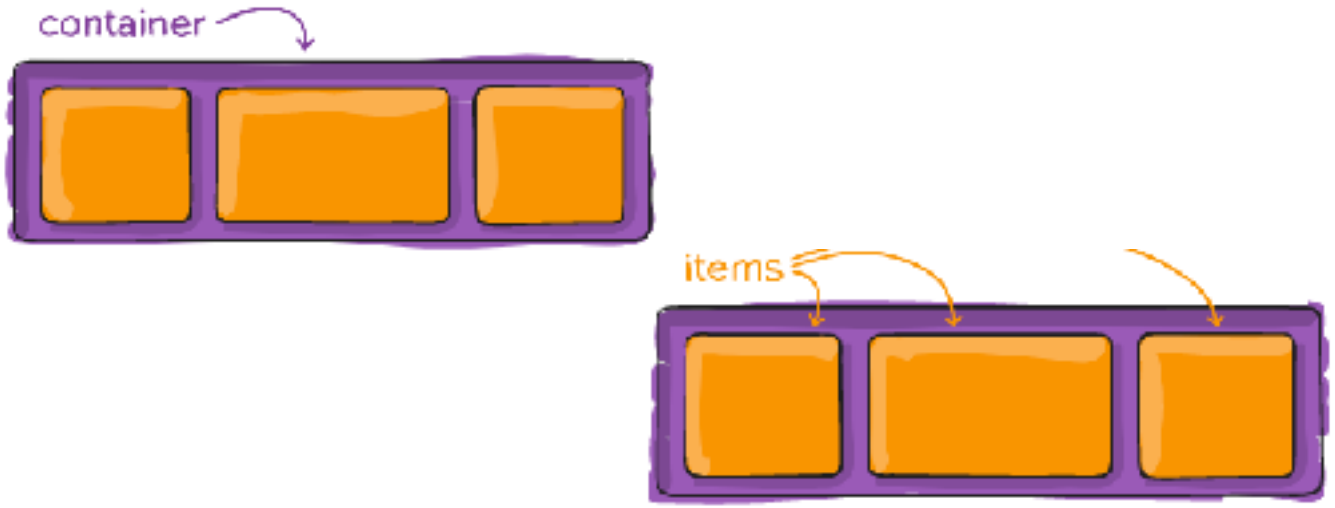
Fem positionsmodeller

- Position bestämmer hur ett block ska placeras ut på sidan i förhållande till andra
    - **static** (default): placerar ut element i den ordning de förekommer i dokumentet. Properties top, right, bottom, left och z-index har ingen effekt
    - **relative**: liknar static men tillåter att properties top, right, bottom, left och z-index används för att ändra positionen relativt till föregående element.
    - **absolute**: positionerar elementet i förhållande till närmsta icke-statiska förälder. Properties top, right, bottom, left och z-index används för att ändra positionen.
    - **fixed**: beter sig på samma sätt som absolute men i förhållande till webbläsarfönstret (viewport)
    - **sticky**: beter sig som relative tills dess att en specificerad offset uppfylls varefter den beter sig som fixed.
- Exempel: https://jsfiddle.net/4nepm12L/46/

Layout - Flexbox

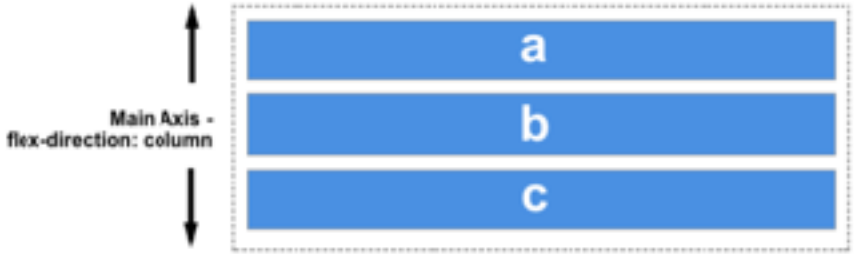
Flexbox-layout

- Flera properties används tillsammans
- Vissa properties sätts på container-nivån (flex container)
- Vissa properties sätts på elementen i containern (flex items)

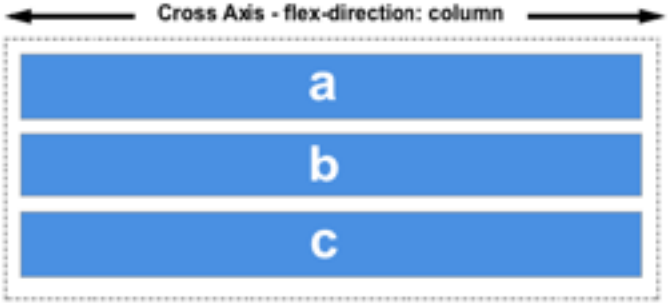


Huvudaxel och korsande axel

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.

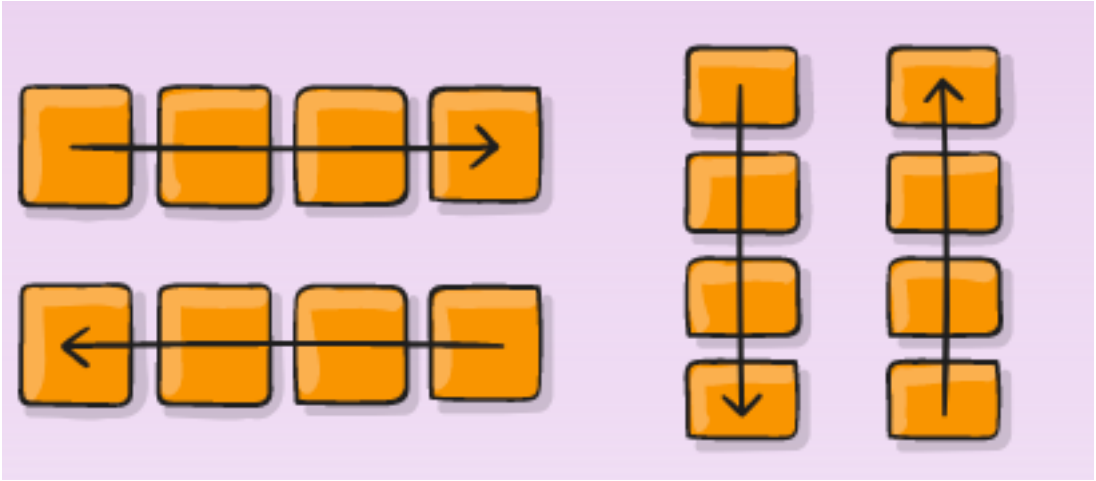


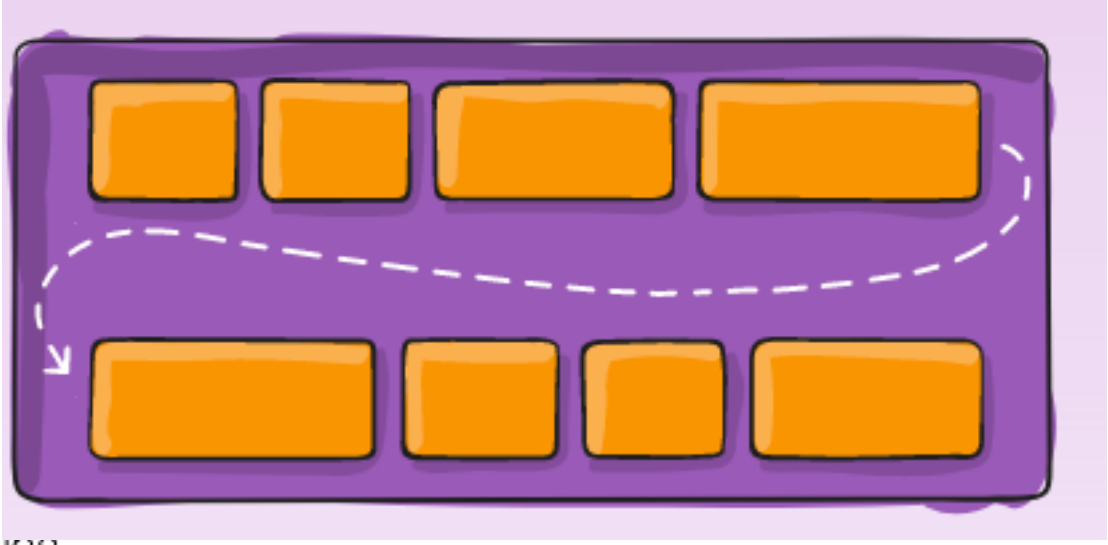
Några properties på container-nivå

- 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>

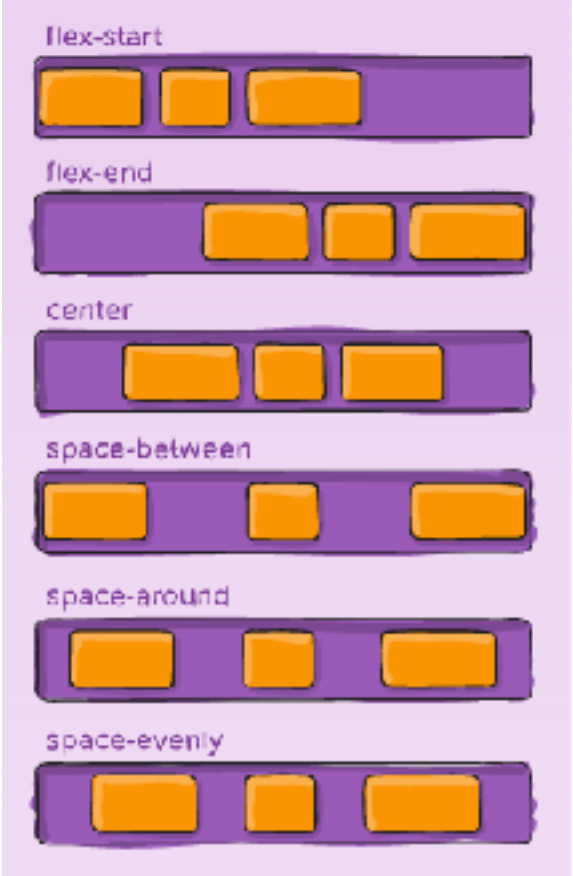
flex-direction and flex-wrap

-

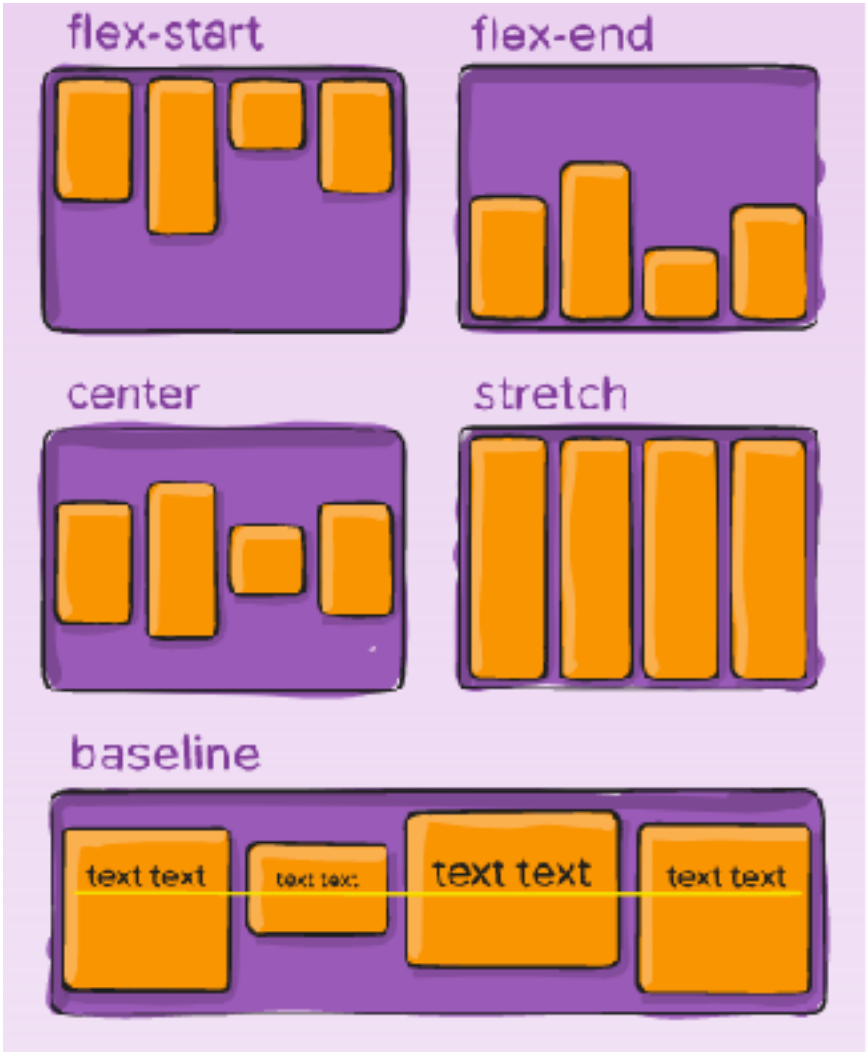


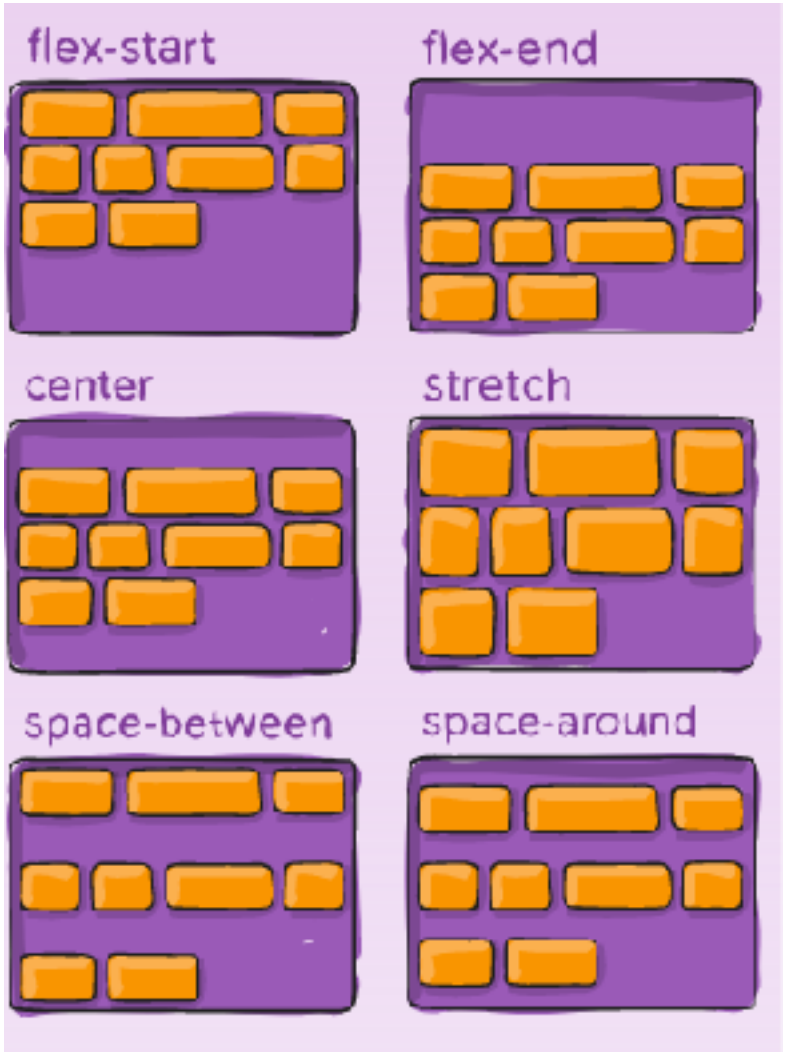


justify-content



align-items and align-content

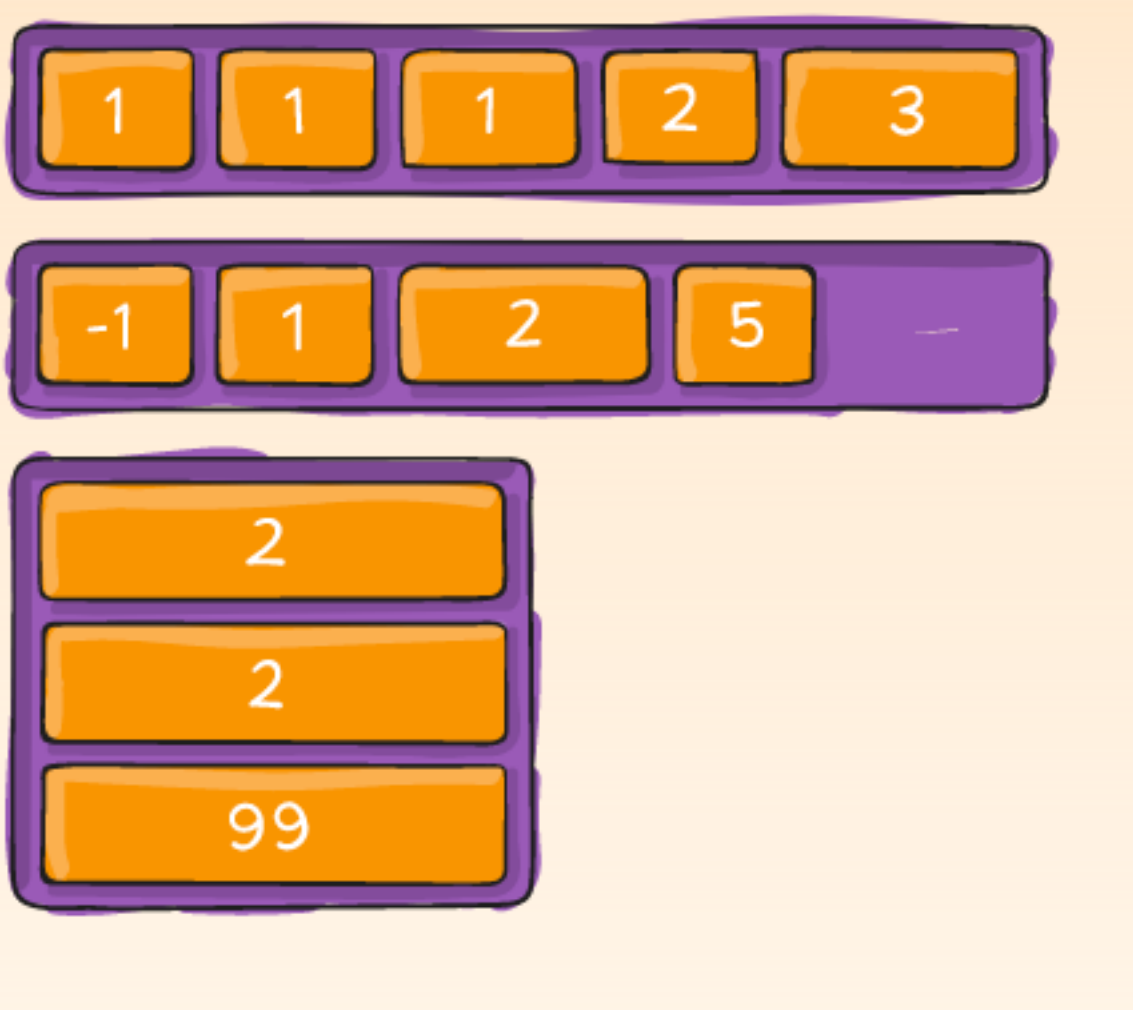




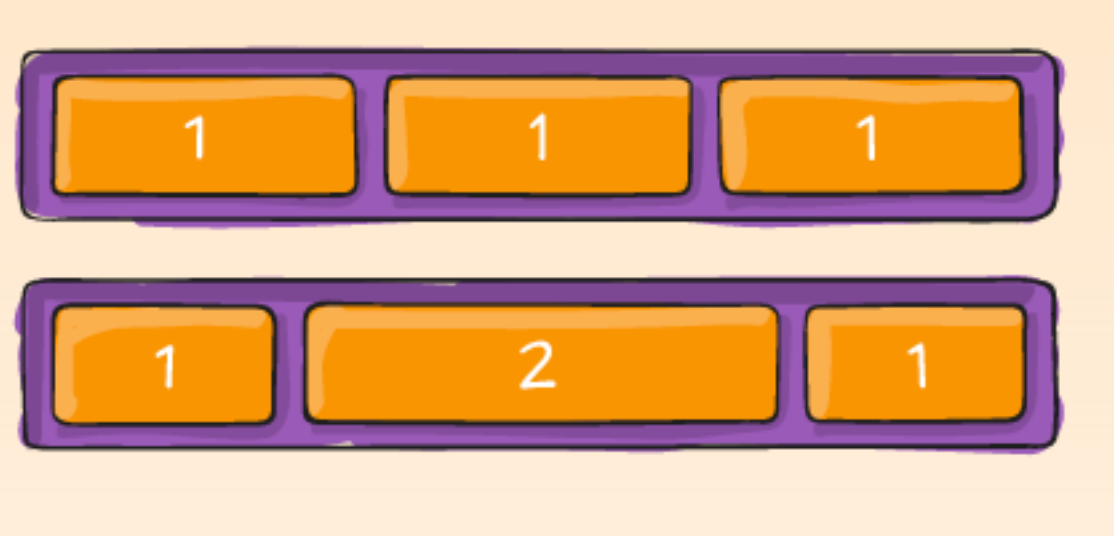
Properties på flex items

- 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

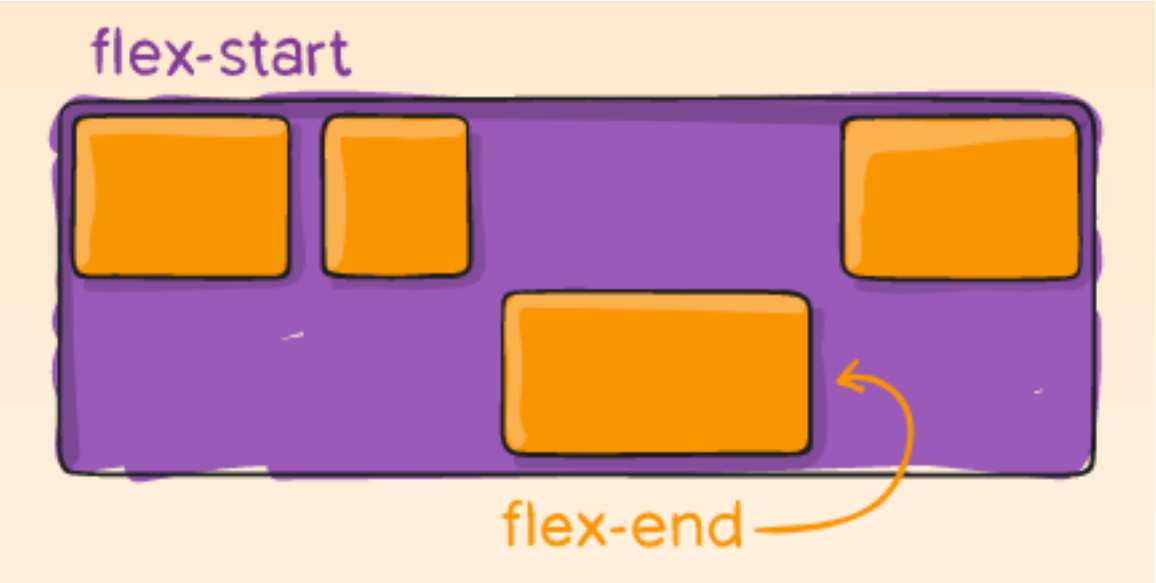
order



flex-grow and flex-shrink



align-self



Flexbox Froggy

https://flexboxfroggy.com/

Grid Garden

https://cssgridgarden.com/

Livekodning

JavaScript

JavaScript (ES6)

- Mer eller mindre fullständigt stöd i alla moderna webbläsare
    - Chrome, Firefox, Safari, Opera, Edge, Brave, …
- Finns senare versioner men med mer begränsat stöd
- JavaScript körs (generellt sett) endast i en tråd
- Objektorienterad programmering stöds men kostar mer minne då varje objekt definierar sina egna funktioner (prototyping)

index.html

```html
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>A document</title>
    <script src="script.js">
  </head>
  <body>
    <h1>The Heading</h1>
    <p>The paragraph</p>
  <body>
</html>
```

script.js

```js
window.onload = () => {
  console.log('Page loaded!');
  let names = ['Bob', 'Dylan', 'Woodie',
               'Guthrie'];
  for(let name of names){
    hello(name);
  }
}

function hello(name){
  console.log(`Hello ${name}!`);
}
```

Viktigaste tillskotten i ES6

- arrow functions
- let + const
- iterators + for..of
- promises
- template strings
- classes
- modules

Syntax

```
// Deklarera en global variabel
let foo = 'Hello';

// En funktion
function helloWorld(repeat) {
  // Deklarera en lokal variabel
  let bar = 'World!';

  // Logga till konsol
  console.log(foo);

  let greeting = '';
  // for-loop
  for(let i=0; i<repeat; i++){
    // Slå samman strängarna och lägg till i greeting
    greeting += foo + ' ' + bar;
  }
  return greeting;
}

// funktionsanrop
helloWorld(12);
```

Callback-funktioner

- Callback-funktioner är ett av de viktigaste koncepten i JavaScript
- Lämnar över ansvaret för att fånga upp data och event till den kallade funktionen
- Utgör en stor del av JavaScript och interaktionen med tredjeparts-bibliotek
- *”När du har laddat ned bilden så gör det här.”*

- Vi kommer gå in närmare på callbacks och asynkrona anrop i nästa föreläsning

Tilldelningar och scope

- Olika sätt att deklarera variabler
  - **var** (ES5)
  - **let** (ES6)
  - **const** (ES6)
- Skillnaden ligger i scope
  - **var** har ett scope definierat av närmaste funktion
  - **let** och **const** har ett scope definierat av närmaste block
- Exempel: https://jsfiddle.net/08frseu1/43/