TEMPLATES!

LOGISTICS

- Absolute and relative paths
- JavaScript templates and tagged template literals
- In-class templates activity
- MP3 work time

ABSOLUTE PATHS

Starts with a slash (`/`) which means the *root directory*

```
mp0/
    index.html
index.html
common.css
```

In `mp0/index.html`, `<a href="/">home</a>` will automatically link to whatever `index.html` is in the root directory.

RELATIVE PATHS

- Do not start with a slash, e.g. `folder/filename.txt`
  - A period (`.`) means “the current directory”: `./folder.txt`
  - Two periods (`..`) means “the directory above me”: `../filename.txt`

```
mp0/
    index.html
index.html
common.css
```

In `mp0/index.html`, `<a href="../">home</a>` will also link to whatever `index.html` is in the root directory.

USING `index.html` PROPERLY

If you name your file `index.html` in a subfolder, you don’t need to include the filename in the path!

```
mp0/
    index.html
index.html
```

In the root `index.html`, `<a href="mp0/">MP0</a>` will automatically link to whatever `index.html` is in the `mp0` folder.

EXAMPLE FOLDER STRUCTURE

```
mp0/
    index.html
ex/
    index.html
index.html
nav.js
common.css
```

EXAMPLE FOLDER STRUCTURE

```
mp0/
    index.html
ex/
    index.html
index.html
nav.js
common.css
```

How do I import `common.css` to `index.html`? In other words, what is the path that should go in `<link rel="stylesheet" href="???" />`

FROM ROOT

In `index.html`:

- `/common.css`
- `common.css`
- `./common.css`

FROM SUBFOLDER

```
mp0/
    index.html
ex/
    index.html
index.html
nav.js
common.css
```

FROM SUBFOLDER

```
mp0/
    index.html
ex/
    index.html
index.html
nav.js
common.css
```

How do I import `common.css` to `ex/index.html`? In other words, what is the path that should go in `<link rel="stylesheet" href="???" />`

FROM SUBFOLDER

In `ex/index.html`:

- `../common.css`
- `/common.css`
- `./../common.css`

WHY TEMPLATE LITERALS

- Often used as an alternative to string concatenation
- Copying and pasting things gets annoying!
- Many things to maintain and change!

TEMPLATE LITERAL SYNTAX

```
// Delimited with backticks
`string text`;

// Multi-line strings
`string text line 1
string text line 2`;

// Inputs with the ${} syntax
`string text ${expression} string text`;

// Tag functions can perform additional operations
tagFunction`string text ${expression} string text`;
```

A SIMPLE TEMPLATE

```
const name = "hannah";
console.log(`Hi, I'm ${name}!`);
```

EXPRESSIONS IN TEMPLATES

```
const a = 5;
const b = 10;
console.log("Fifteen is " + (a + b) + " and not " + (2 * a + b) +
        ".");

const a = 5;
const b = 10;
console.log(`Fifteen is ${a + b} and not ${2 * a + b}.`);
```

TAGGED TEMPLATE LITERALS

- Templates are not only used for string interpolation
- Template literals can be *tagged* using a tag function
- Tag functions perform whatever operations you want!

TAG FUNCTIONS

```
const person = "Mike";
const age = 28;

function myTag(strings, personExp, ageExp) {
  const str0 = strings[0]; // "That "
  const str1 = strings[1]; // " is a "
  const str2 = strings[2]; // "."

  const ageStr = ageExp > 99 ? "centenarian" : "youngster";

  // We can even return a string built using a template literal
  return `${str0}${personExp}${str1}${ageStr}${str2}`;
}

const output = myTag`That ${person} is a ${age}.`;

console.log(output);
```

`lit-html` TEMPLATES

- `lit-html` is a package for writing tagged template literals
- Two main exports: `html` and `render`:
  - **`html`**: A JavaScript template tag
  - **`render`**: : Function to render an HTML template to a DOM container
- Combines the familiarity of writing HTML with the power of JavaScript!

HTML TEMPLATE EXAMPLE

```
// Importing the html template tag and render function
import { html, render } from "lit-html";

// This is a lit-html template function. It returns
// a lit-html template.
const helloTemplate = (name) => html`<div>Hello ${name}!</div>`;

// This renders <div>Hello Steve!</div> to the document body
render(helloTemplate("Steve"), document.body);

// This updates to <div>Hello Kevin!</div>, but only updates
// the ${name} part
render(helloTemplate("Kevin"), document.body);
```

DYNAMIC DATA

```
import { html, render } from "lit-html";

// Define a template function
const myTemplate = (name) => html`<div>Hello ${name}</div>`;

// Render the template with some data
render(myTemplate("earth"), document.body);

// ... Later on ...
// Render the template with different data
render(myTemplate("mars"), document.body);
```

THE LIT-HTML PACKAGE

Install with npm:

```
npm install lit-html
```

```
// javascript
import { html, render } from "lit-html";
```

Import from cdn:

```
// javascript
import { html, render } from "https://unpkg.com/lit-html?module";
```