Topic 8: Cascading Style Sheets

Display Property

Lecture Contents

- Evolution
- CSS1 to CSS2.1 features
  - **`inline`**, **`block`**, **`inline-block`**
- CSS3 features
  - **`flex`**, **`grid`**

Evolution

- Many features of web design evolved over time
  - To understand why we have certain features, we often need to understand this evolution.
  - The web is currently using CSS version 3 (CSS3)

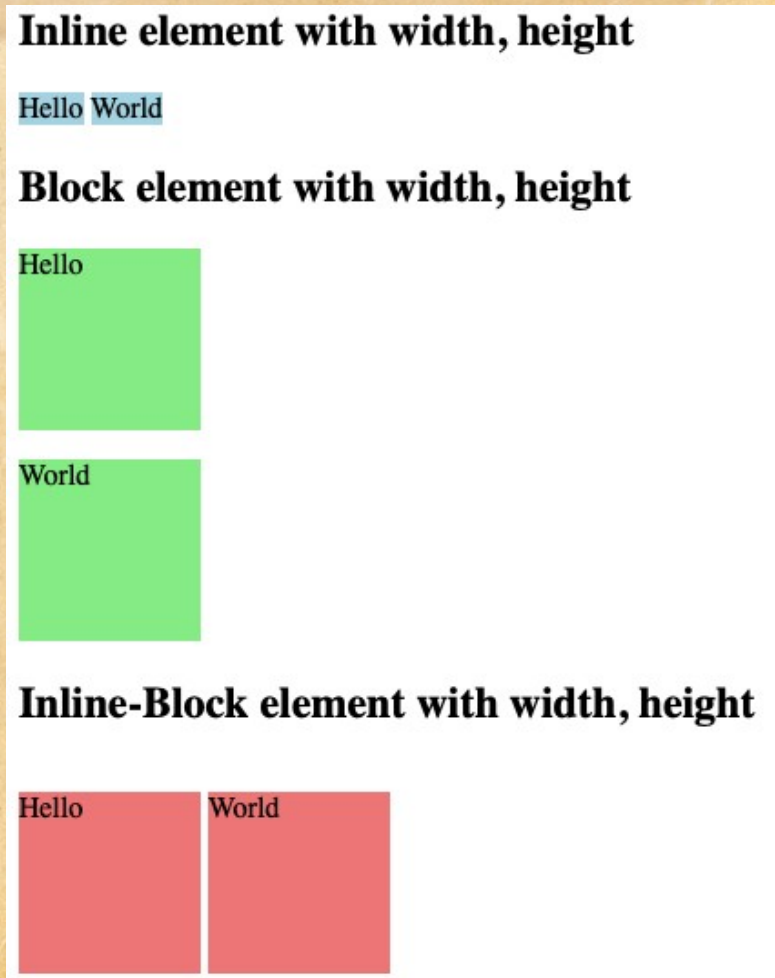
Evolution of the `display` property

- CSS1
  - **`block`**: start on a new line; take up the full width available
  - **`inline`**: continue the line; take up only as much width as necessary
- CSS2
  - **`none`**: hides and no effect on layout (unlike `hidden` property!)
- CSS2.1
  - **`inline-block`**: continues the line; takes up only as much width as necessary; add features unavailable with `inline`
    - set `width` and `height` respected
    - common use case: html list → menu displayed horizontally

Evolution of the `display` property

```
<h2>Inline element with width, height</h2>
<p class="inline">Hello</p>
<p class="inline">World</p>

<h2>Block element with width, height</h2>
<p class="block">Hello</p>
<p class="block">World</p>

<h2>Inline-Block element with width, height</h2>
<p class="inline-block">Hello</p>
<p class="inline-block">World</p>

</body>
</html>
```

```
.inline {
  display: inline;
  width: 100px;
  height: 100px;
  background-color: #add8e6;
}

.block {
  display: block;
  width: 100px;
  height: 100px;
  background-color: #90ee90;
}

.inline-block {
  display: inline-block;
  width: 100px;
  height: 100px;
  background-color: #f08080;
}
```



Evolution of the `display` property

- Examples of elements that are `inline` by default
  - **`span`**, `a`, `img`, …
- Examples of elements that are `block` by default
  - **`div`**, `h1`, `p`, `form`, `header`, `footer`, …

each highlighted example is the generic tag

Evolution of the `display` property

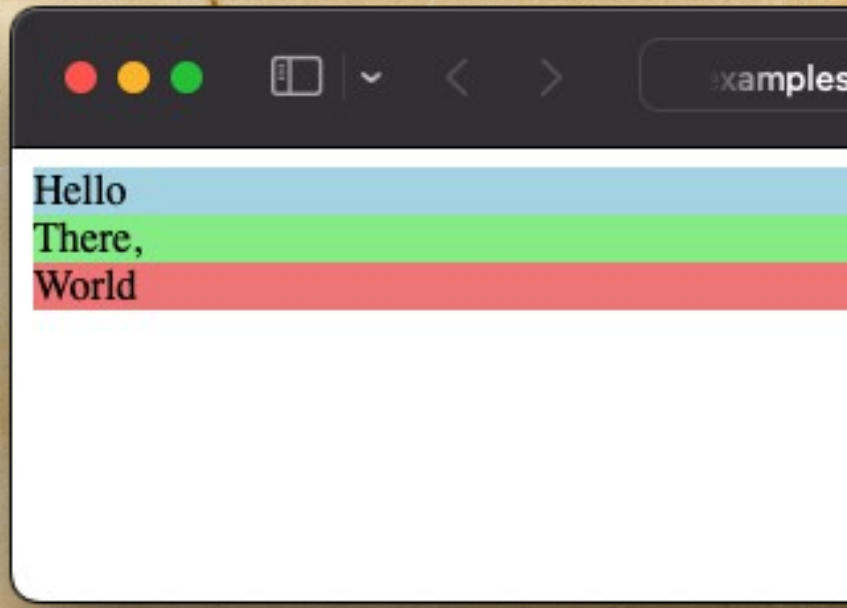
- CSS3
  - **`contents`**: container disappears; children become children of next level up in the DOM.
  - **`flex`**: efficient layout, alignment, distribution of space in a container, even when child sizing is unknown or dynamic
  - **`grid`**: two-dimensional arrangement using rows and columns

CSS `display: flex`

- Applied to the *parent* of the elements to be styled
-

CSS display: flex

```
<div class="flex-container">
  <div class="flexbox-item flexbox1">Hello</div>
  <div class="flexbox-item flexbox2">There,</div>
  <div class="flexbox-item flexbox3">World</div>
</div>
```

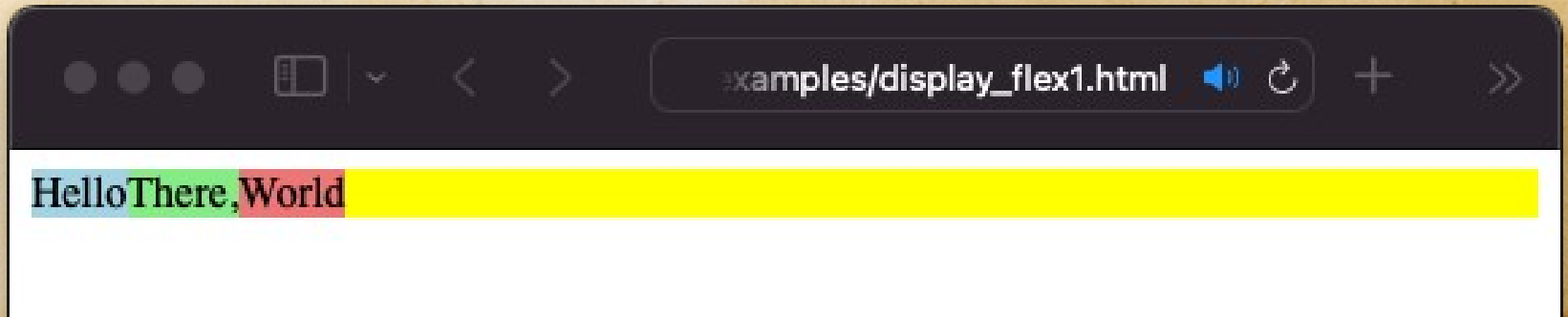



```
.flexbox1 {
    background-color: #add8e6;
}
.flexbox2 {
    background-color: #90ee90;
}
.flexbox3 {
    background-color: #f08080;
}
```

CSS display: flex

```
<div class="flex-container">
  <div class="flexbox-item flexbox1">Hello</div>
  <div class="flexbox-item flexbox2">There,</div>
  <div class="flexbox-item flexbox3">World</div>
</div>
```

```
.flex-container {
    display:flex;
    background-color: yellow;
    /* justify-content: flex-start */
}
```



CSS display: flex

```
<div class="flex-container">
  <div class="flexbox-item flexbox1">Hello</div>
  <div class="flexbox-item flexbox2">There,</div>
  <div class="flexbox-item flexbox3">World</div>
</div>
```

```
.flex-container {
    display:flex;
    background-color: yellow;
    justify-content: center;
}
```

CSS display: flex

```
<div class="flex-container">
  <div class="flexbox-item flexbox1">Hello</div>
  <div class="flexbox-item flexbox2">There,</div>
  <div class="flexbox-item flexbox3">World</div>
</div>
```

```
.flex-container {
    display:flex;
    background-color: yellow;
    justify-content: space-between;
}
```

CSS display: flex

```
<div class="flex-container">
  <div class="flexbox-item flexbox1">Hello</div>
  <div class="flexbox-item flexbox2">There,</div>
  <div class="flexbox-item flexbox3">World</div>
</div>
```

```
.flex-container {
    display:flex;
    background-color: yellow;
    justify-content: space-around;
}
```

CSS display: flex

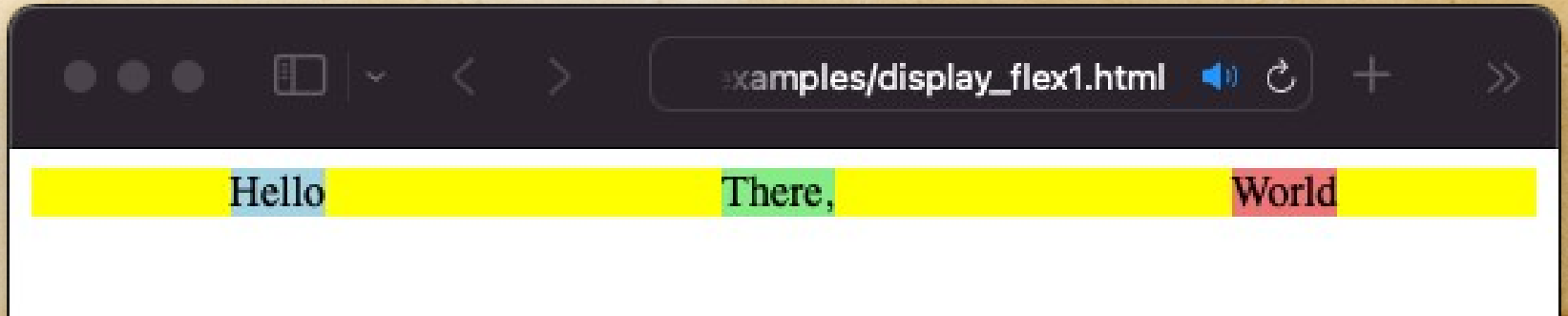
```
<div class="flex-container">
  <div class="flexbox-item flexbox1">Hello</div>
  <div class="flexbox-item flexbox2">There,</div>
  <div class="flexbox-item flexbox3">World</div>
</div>
```

```
.flex-container {
    display:flex;
    background-color: yellow;
    justify-content: space-around;
}
```

CSS display: flex

```
<div class="flex-container">
  <div class="flexbox-item flexbox1">Hello</div>
  <div class="flexbox-item flexbox2">There,</div>
  <div class="flexbox-item flexbox3">World</div>
</div>
```

```
.flexbox1 {
}
.flexbox2 {
    max-height: 30px;
}
.flexbox3 {
}
```



CSS display: flex

```
<div class="flex-container">
  <div class="flexbox-item flexbox1">Hello</div>
  <div class="flexbox-item flexbox2">There,</div>
  <div class="flexbox-item flexbox3">World</div>
</div>
```

```
.flexbox1 {
    min-height: 60px;
}
.flexbox2 {
    max-height: 30px;
}
.flexbox3 {
}
```

CSS display: flex

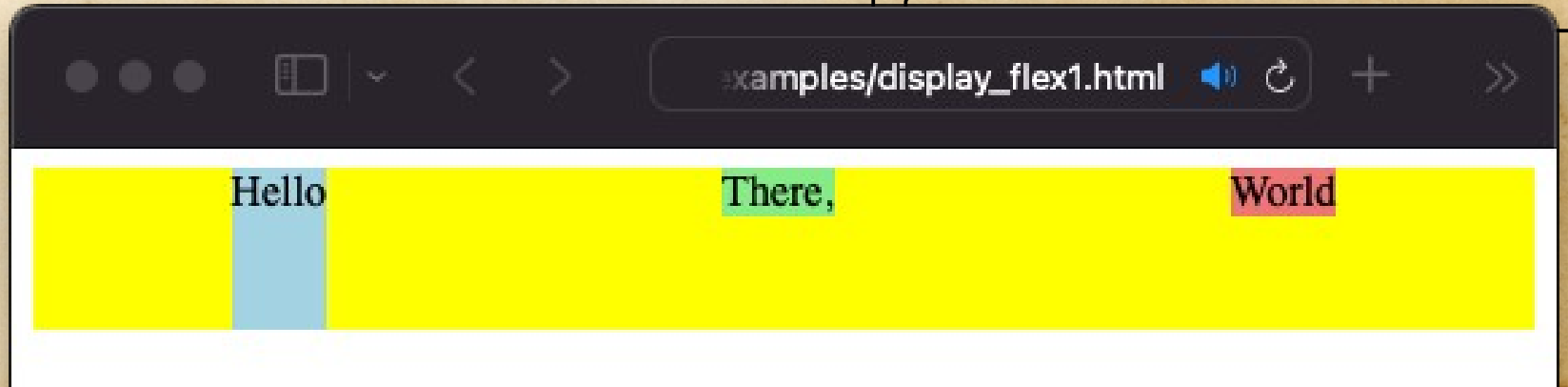
```
<div class="flex-container">
  <div class="flexbox-item flexbox1">Hello</div>
  <div class="flexbox-item flexbox2">There,</div>
  <div class="flexbox-item flexbox3">World</div>
</div>
```

```
.flexbox-container {
    /* align-items: stretch; */
}
.flexbox1 {
    min-height: 60px;
}
.flexbox2 {
    max-height: 30px;
}
.flexbox3 {
}
```

CSS display: flex

```
<div class="flex-container">
  <div class="flexbox-item flexbox1">Hello</div>
  <div class="flexbox-item flexbox2">There,</div>
  <div class="flexbox-item flexbox3">World</div>
</div>
```

```
.flexbox-container {
    align-items: flex-start;
}
.flexbox1 {
    min-height: 60px;
}
.flexbox2 {
    max-height: 30px;
}
.flexbox3 {
}
```



CSS display: flex

```
<div class="flex-container">
  <div class="flexbox-item flexbox1">Hello</div>
  <div class="flexbox-item flexbox2">There,</div>
  <div class="flexbox-item flexbox3">World</div>
</div>
```

```
.flexbox-container {
    align-items: flex-end;
}
.flexbox1 {
    min-height: 60px;
}
.flexbox2 {
    max-height: 30px;
}
.flexbox3 {
}
```

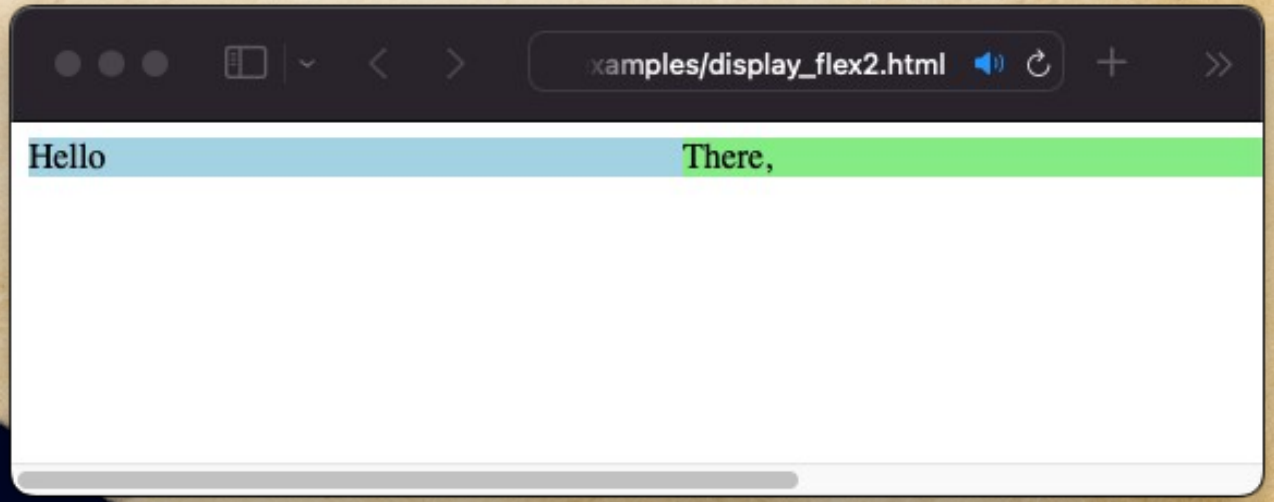
CSS display: flex

```
<div class="flex-container">
  <div class="flexbox-item flexbox1">Hello</div>
  <div class="flexbox-item flexbox2">There,</div>
  <div class="flexbox-item flexbox3">World</div>
</div>
```

```
.flexbox-container {
    align-items: center;
}
.flexbox1 {
    min-height: 60px;
}
.flexbox2 {
    max-height: 30px;
}
.flexbox3 {
}
```

CSS display: flex

```
<div class="flex-container">
  <div class="flexbox-item flexbox1">Hello</div>
  <div class="flexbox-item flexbox2">There,</div>
  <div class="flexbox-item flexbox3">World</div>
</div>
```




```
.flex-container {
    /* flex-wrap: nowrap; */
}
.flexbox1 {
    min-width: 300px;
}
.flexbox2 {
    min-width: 300px;
}
.flexbox3 {
    min-width: 300px;
}
```

CSS display: flex

```
<div class="flex-container">
  <div class="flexbox-item flexbox1">Hello</div>
  <div class="flexbox-item flexbox2">There,</div>
  <div class="flexbox-item flexbox3">World</div>
</div>
```


```
.flex-container {
    flex-wrap: wrap;
}
.flexbox1 {
    min-width: 300px;
}
.flexbox2 {
    min-width: 300px;
}
.flexbox3 {
    min-width: 300px;
}
```

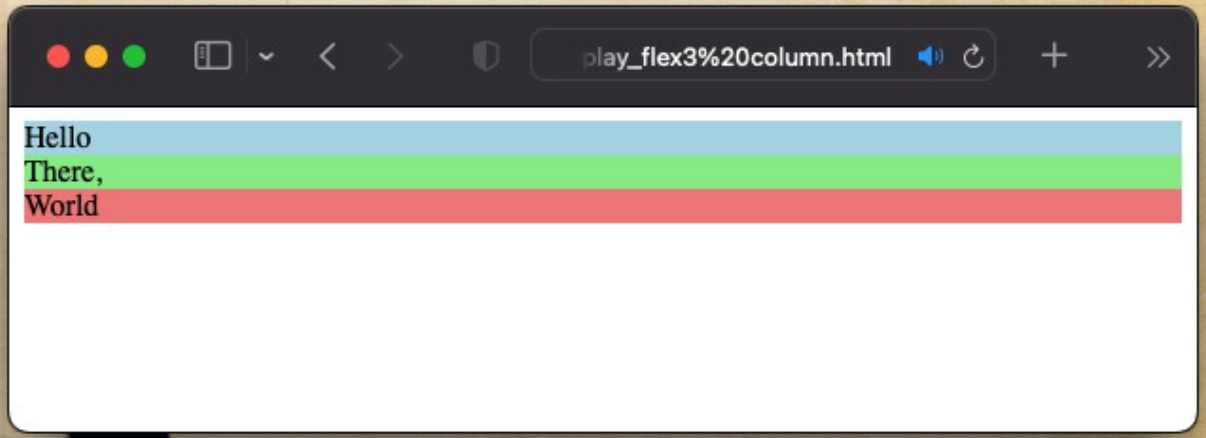
CSS display: flex

```
<div class="flex-container">
  <div class="flexbox-item flexbox1">Hello</div>
  <div class="flexbox-item flexbox2">There,</div>
  <div class="flexbox-item flexbox3">World</div>
</div>
```


```
.flex-container {
    flex-wrap: wrap-reverse;
}
.flexbox1 {
    min-width: 300px;
}
.flexbox2 {
    min-width: 300px;
}
.flexbox3 {
    min-width: 300px;
}
```

CSS display: flex

```
<div class="flex-container">
  <div class="flexbox-item flexbox1">Hello</div>
  <div class="flexbox-item flexbox2">There,</div>
  <div class="flexbox-item flexbox3">World</div>
</div>
```




```
.flex-container {
    display:flex;
    background-color: yellow;
    flex-direction: column;
    /* align-items: stretch */
}
.flexbox1 {
    background-color: #add8e6;
}
.flexbox2 {
    background-color: #90ee90;
}
.flexbox3 {
    background-color: #f08080;
}
```

CSS display: flex

```
<div class="flex-container">
  <div class="flexbox-item flexbox1">Hello</div>
  <div class="flexbox-item flexbox2">There,</div>
  <div class="flexbox-item flexbox3">World</div>
</div>
```


```
.flex-container {
    display:flex;
    background-color: yellow;
    flex-direction: column;
    align-items: flex-start;
}
.flexbox1 {
    background-color: #add8e6;
}
.flexbox2 {
    background-color: #90ee90;
}
.flexbox3 {
    background-color: #f08080;
}
```

CSS display: flex

```
<div class="flex-container">
  <div class="flexbox-item flexbox1">Hello</div>
  <div class="flexbox-item flexbox2">There,</div>
  <div class="flexbox-item flexbox3">World</div>
</div>
```


```
.flex-container {
    display:flex;
    background-color: yellow;
    flex-direction: column;
    align-items: center;
}
.flexbox1 {
    background-color: #add8e6;
}
.flexbox2 {
    background-color: #90ee90;
}
.flexbox3 {
    background-color: #f08080;
}
```

CSS `display: flex`

```
<div class="flex-container">
  <div class="flexbox-item flexbox1">Hello</div>
  <div class="flexbox-item flexbox2">There,</div>
  <div class="flexbox-item flexbox3">World</div>
</div>
```

```
.flexbox-container {
    /* flex-direction: row; */
    align-items: center;
}
.flexbox1 {
    min-height: 60px;
}
.flexbox2 {
    max-height: 30px;
}
.flexbox3 {
```

Note the **`align-items`** is in relation to the major axis – which is the x-axis for **`row`** and y-axis for **`column`**

CSS `display: flex`

- For the flex container:
  - `flex-shrink`: zero means do not shrink below initial size; positive number shrink value relative to other flex items; default value 1.
  - `flex-grow`: grow factor relative to other flex items; default value 0, meaning it will not grow.
  - `flex-basis`: initial size of flex item before shrink/grow; default value is the contents of the box (or set box sizing); if set to zero, the growth calculation will not start from initial size

CSS `display: flex`

- For the flex container:
  - `order`: change the order to display the flex box items.
    - Perhaps not the best way to do things: change the HTML?
    - Doesn’t change the order for screen readers?

CSS `display: grid`

- Coming Soon!

Topic 8: Cascading Style Sheets

Display Property