

CSS

Shan-Hung Wu

CS, NTHU

CSS Zen Garden

Outline

- The Basics
- Selectors
- Layout
- Stacking Order

Outline

- The Basics
- Selectors
- Layout
- Stacking Order

Grammar

```
selector {  
    property: value;  
}
```

Example

```
/* for all h1's */  
h1 {  
    color: red;  
    font-size: 56px;  
}  
  
/* for all img's */  
img {  
    border-color: blue;  
    border-width: 3px;  
}
```

Color & Background

```
h1 {  
    color: #4b0082;  
    background: #95a5a6;  
}
```

```
h1 {  
    color: rgba(11, 99, 150, 0.3);  
}
```

More Background Properties

```
body {  
    background: #000 /* color */  
               url("img.png") /* image */  
               no-repeat /* repeat */  
               fixed /*attachment */  
               right top; /*position */  
}
```

- CSS3

```
body {  
    background-size: cover;  
}
```


CSS Properties

- Background:
 - E.g., background-color, background-image, etc.
- Text:
 - E.g., color, text-align, text-decoration, etc.
- Font:
 - E.g., font-family, font-size, font-style, font-weight, etc.
- List & table:
 - E.g., list-style-type, list-style-image, vertical-align, etc.
- Layout:
 - E.g., width, height, border, padding, margin, display, visibility, float, position, etc.
- See [a list here](#)

Native Font Stack

```
body {  
  font-family:  
    /* Safari for OS X and iOS */  
    -apple-system,  
    /* Chrome >= 56 for OS X , Windows, Linux and Android */  
    system-ui,  
    /* Chrome < 56 for OS X */  
    BlinkMacSystemFont,  
    /* Windows */  
    "Segoe UI",  
    /* Android */  
    "Roboto",  
    /* Basic web fallback */  
    "Helvetica Neue", Arial, sans-serif;  
}
```

Google Fonts

Outline

- The Basics
- **Selectors**
- Layout
- Stacking Order

Selectors

```
<ul id="todo-list">  
  <li class="done">TODO 1</li>  
  <li>TODO 2</li>  
  <li class="done">TODO 3</li>  
</ul>
```

```
li {                                /* element/tag selector */  
  font-weight: bold;  
}  
#todo-list {                       /* ID selector */  
  background-color: gray;  
}  
.done {                           /* class selector */  
  text-decoration: line-through;  
}
```

Chrome Inspector

More Selectors

```
/* composition */  
#todo-list, li.done {...}
```

```
/* descendant selector */  
li a {...}
```

```
/* adjacent selector */  
li.done+li {...}
```

```
/* attribute selector */  
a[href="http://..."] {...}
```

Pseudo Classes & Elements

```
/* pseudo class selector */  
a:hover, a:visited {  
    ...  
}  
li:nth-of-type(3) {  
    ...  
}
```

```
/* pseudo element selector */  
p::first-letter {  
    font-size: 200%;  
}  
h1::before {  
    content: url(image.gif);  
}
```

- More [selector examples](#)

Inheritance

- Most style properties of an element will be *inherited* by its descendants
 - E.g., font, color, text-align, etc.
- Common exceptions are those related to the box model
 - E.g., height, width, border-width, etc.
- Check [this reference](#) to see if a property is inheritable

Cascading

- Final properties gotten by an element are ***cascaded*** from all applicable rules

```
<body>
  <ul id="todo-list">
    <li>...</li>
    <li class="done">
      TODO
    </li>
  </ul>
</body>
```

```
body {
  color: gray;
}
#todo-list {
  font-weight: bold;
}
#todo-list li {
  color: red;
}
li.done {
  text-decoration:
    line-through;
}
```

How to Resolve Conflicts?

1. By importance

```
body {  
    color: gray !important;  
}
```

2. By *specificity*

Example	# ID Selectors	# Class Selectors	# Type Selectors
ul	0	0	1
ul li.done	0	1	2
#sec-2 ul li (wins)	1	0	2

3. By source order

– Rules written later win

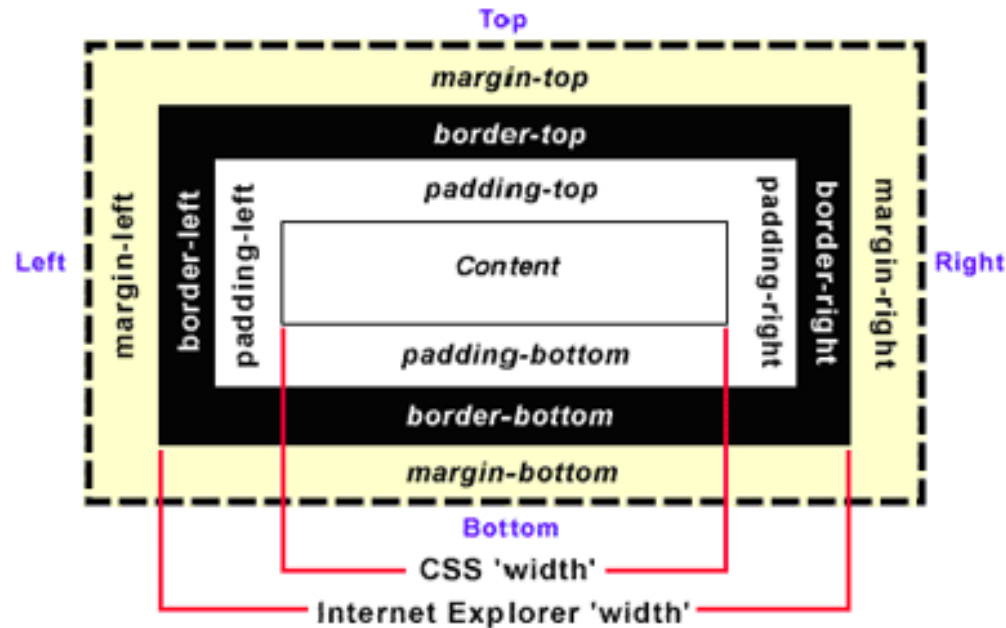
- More about [cascade and inheritance](#)

Outline

- The Basics
- Selectors
- **Layout**
- Stacking Order

Box Model

- Each element is rendered as a box:



- IE6 is problematic unless you give XHTML a DTD

Width and Height

```
div {  
  width: 70%;  
  max-width: 640px;  
  font-size: 16px;  
}
```

- It's a good practice to use rem

```
html {  
  font-size: 16px;  
}  
div {  
  width: 70%;  
  max-width: 40rem;  
  font-size: 1rem;  
}
```

Box Sizing

```
div {  
  width: 50%;  
  padding: 1rem;  
  border: 0.25rem solid blue;  
}
```

- **Box sizing in CSS 3:**

```
* {  
  /* border & padding count into width & height */  
  box-sizing: border-box;  
}
```

Inline vs. Block Boxes

- If an inline box wraps into multiple lines, you cannot set its width
 - Not the case for `<input>` and ``
- Inline boxes reserve space for descender chars, e.g., 'g'
- `` is an inline element, so there is (unwanted) space between its bottom border and container

```
img {  
    display: block; /* or inline */  
}
```


Hiding Elements

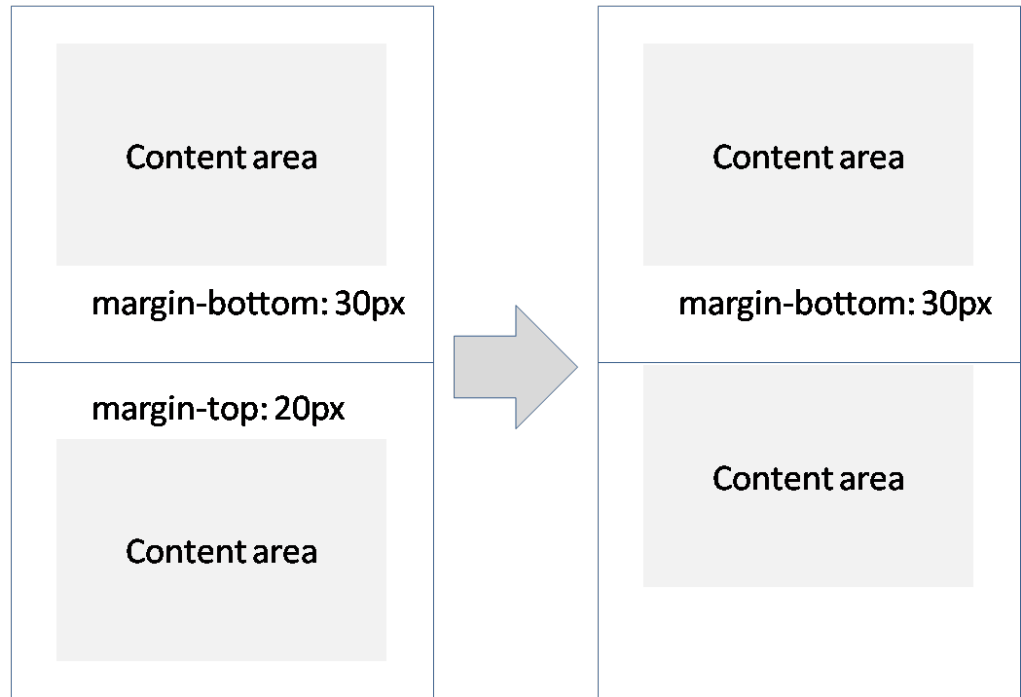
- Method 1: `visibility:hidden`
 - Still occupies space in normal flow
- Method 2: `display:none`
 - Removed from normal flow

Centering Text/Elements

- How to center text/inline elements inside a block element?
 - Add `text-align:center` to the block
- How to center text in an inline element?
 - There is no such an issue
- How to center a block element inside another?
 1. Give inner block a width (otherwise we don't have this issue)
 2. Add `margin-left:auto` and `margin-right:auto` to inner block

Margin Collapsing

- Adjacent margins collapse between
 - Sibling elements
 - Parent and first/last child



HTML Rendering

- The content are rendered following the *normal flow*
 - Block elements are laid out vertically
 - Inline elements are laid out horizontally

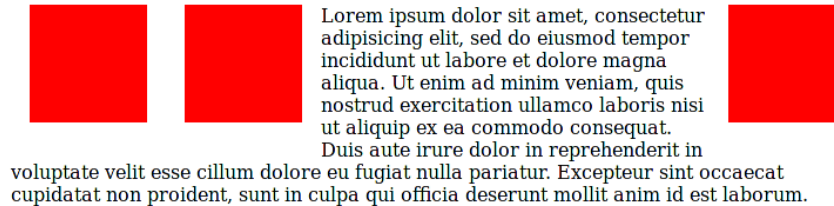
Positioning

- Not positioned: `static` (in normal flow)

```
div {  
    position: relative; /* in normal flow */  
    top: 8px;           /* offset from static position */  
    left: 8px;  
}  
  
div {  
    position: absolute; /* removed from normal flow */  
    top: 8px;           /* offset from positioned ancestor */  
}  
  
div {  
    position: fixed;    /* removed from normal flow */  
    top: 8px;           /* offset from browser window */  
}
```

Floats

```
.elem {  
  float: left;  
}  
  
.container::after {  
  content: '';  
  display: block;  
  clear: both;  
}
```



- Removed from normal flow and stick to the left/right-most side of its container
 - Unless specified, width and height shrink to fit the content

Demo: Photo Gallery

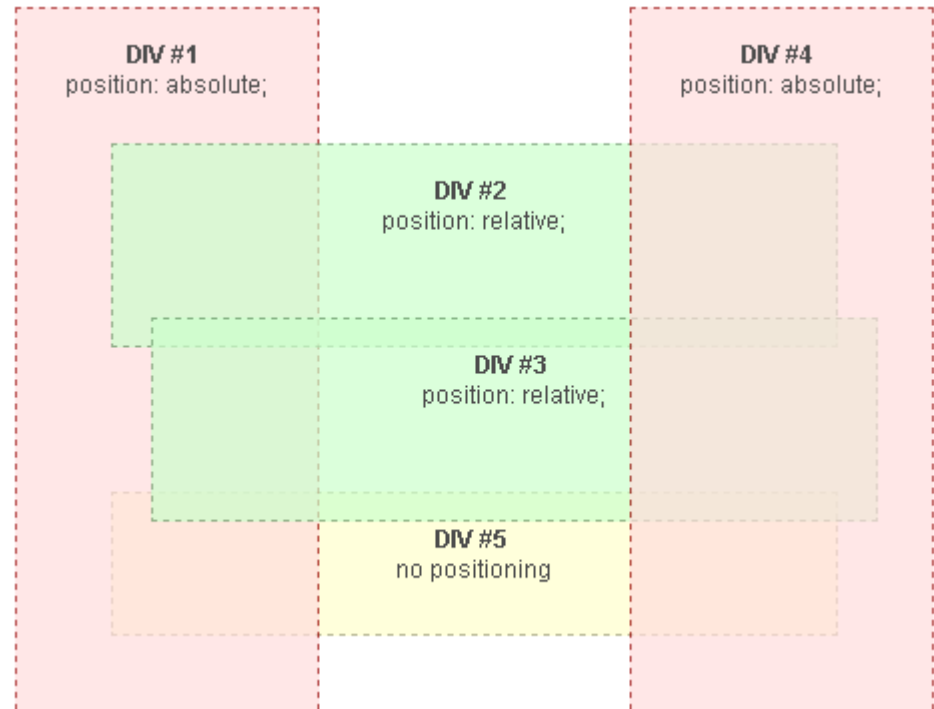
Outline

- The Basics
- Selectors
- Layout
- Stacking Order

Overlapping Elements

- Elements may overlap

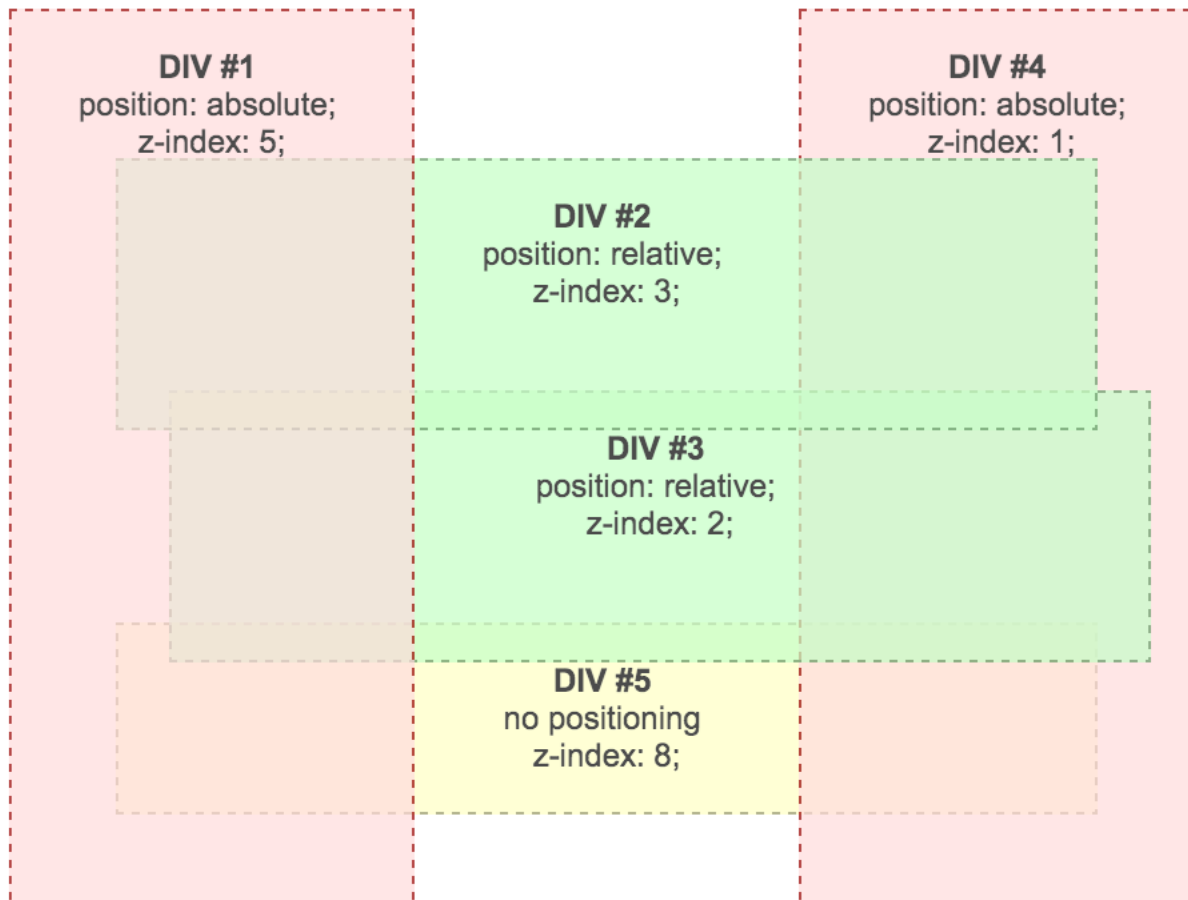
```
<div id="1">...</div>  
<div id="2">...</div>  
<div id="3">...</div>  
<div id="4">...</div>  
<div id="5">...</div>
```



- Which one shows on top?

Z-Index

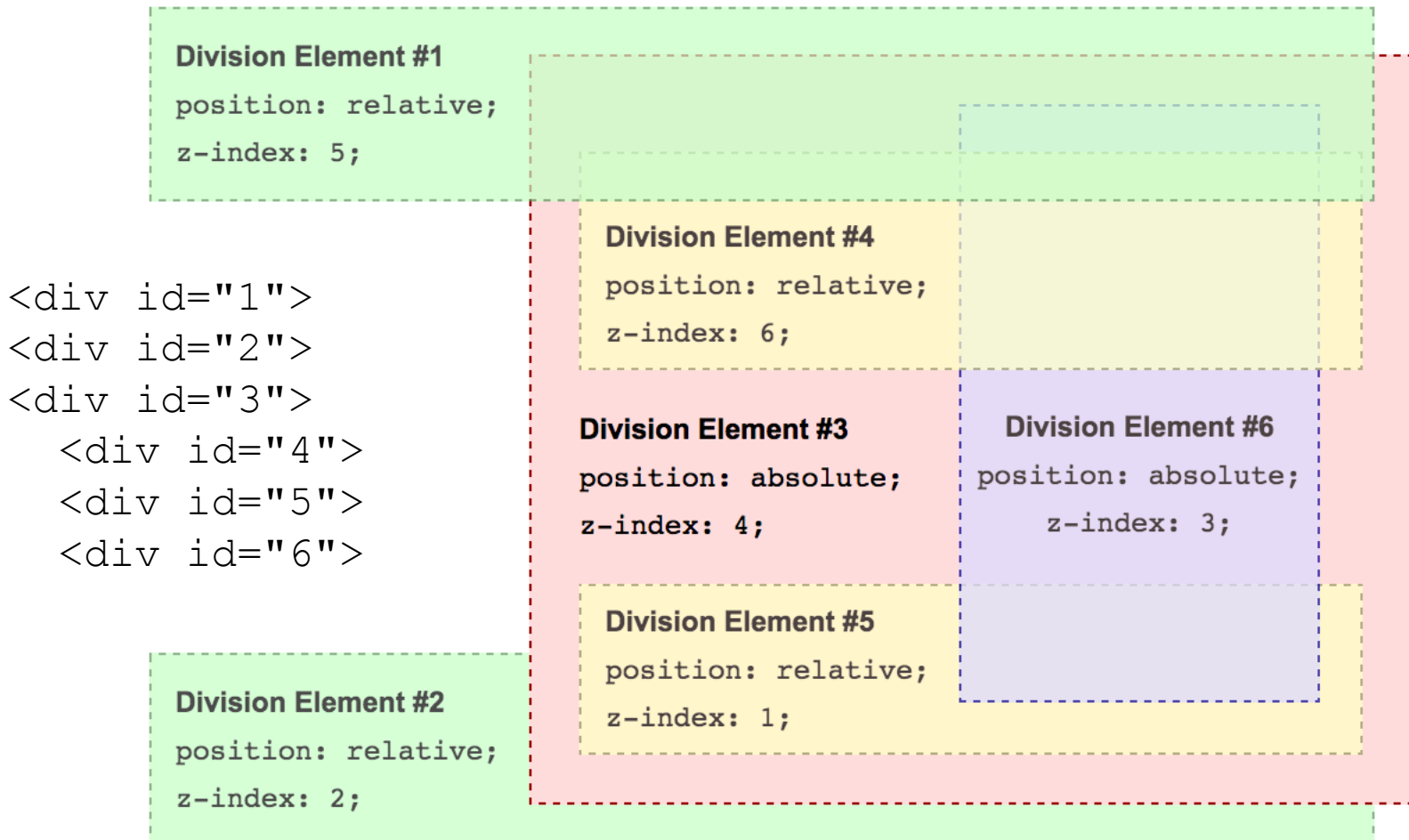
- Default: 0
- Stacking order applies to each layer
- Elements need to be positioned to take effect



How about the order of
descendants?

Stacking Context

- `<html>`, positioned, and non-full opacity elements creates **stacking contexts**
- Z-index is **local** to a stacking context



Assigned Readings

- [CSS tutorial](#)

Exercise

