Interactive Elements

- Class Hierarchy
- CSS Variables
- Basic Interactions: A CSS Approach

Parents & Childs

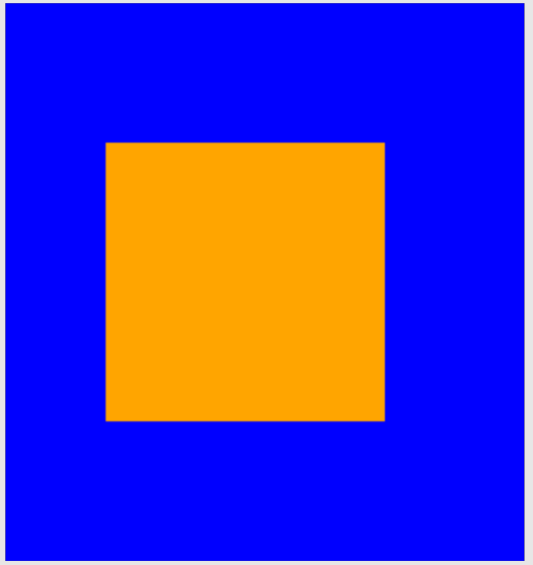
We define the following styles:

```
.parent {
    background: blue;
    width: 200px;
    height: 200px; }
```

```
.child {
    background: orange;
    display: inline-block;
    width: 100px;
    height: 100px;
    margin: 50px; }
```

What is the result when applying it to the following HTML?

```
<div class=parent>
    < div class=child > </div>
</div>
```



Parents & Childs

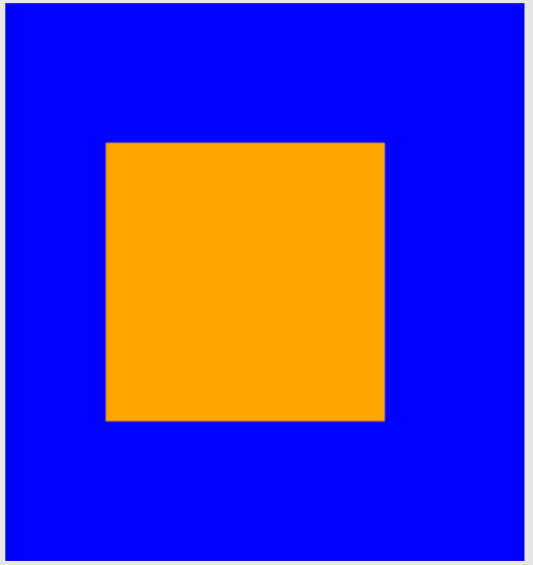
We define the following styles:

```
.parent {
    background: blue;
    width: 200px;
    height: 200px; }
```

```
.parent > .child {
    background: orange;
    display: inline-block;
    width: 100px;
    height: 100px;
    margin: 50px; }
```

What is the result when applying it to the following HTML?

```
<div class=parent>
    < div class=child > </div>
</div>
```



Parents & Childs

We can assign the same properties to multiple classes & IDs.

```
.parent1, .parent2 {
    background: blue;
    width: 200px;
    height: 200px; }
```

'*parent > child*' allows us to reference a class or ID by the immediate parent.

```
.parent1 > .child {
    background: orange;
    display: inline-block;
    width: 100px;
    height: 100px;
    margin: 50px; }
```

```
.parent2 > .child {
    background: green;
    display: inline-block;
    width: 100px;
    height: 100px;
    margin: 50px; }
```

Parents & Childs

We define the following styles:

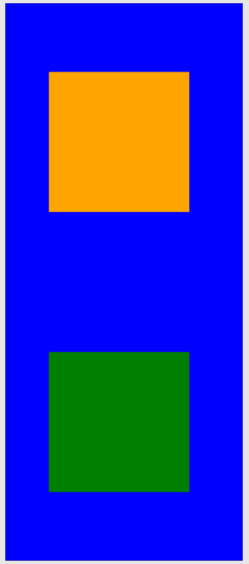
```
.parent1, .parent2 {
    background: blue;
    width: 200px;
    height: 200px; }
```

```
.parent1 > .child {
    background: orange;
    display: inline-block;
    width: 100px;
    height: 100px;
    margin: 50px; }
```

```
.parent2 > .child {
    background: green;
    display: inline-block;
    width: 100px;
    height: 100px;
    margin: 50px; }
```

What is the result when applying it to the following HTML?

```
<div class=parent>
    <div class=child> </div>
</div>
<div class=parent2>
    <div class=child> </div>
</div>
```



Parents & Childs

Looks for class '*child*' as an **immediate** descendant of '*parent*'

```
.parent > .child
```

Looks for class '*child*' as **any** descendant of '*parent*'

```
.parent .child
```

Looks for **any immediate** descendant of '*parent*'

```
.parent > *
```

Looks for **any** descendant of '*parent*'

```
.parent *
```

Parents & Childs

Looks for class '*child*' **immediately after** (not inside) class '*parent*'

```
.parent + .child
```

Looks for class '*child*' **immediately before** (not inside) class '*parent*'

```
.parent ~ .child
```

Looks for **first child** of '*parent*'

```
.parent:first-child
```

Looks for **last child** of '*parent*'

```
.parent:last-child
```

Parents & Childs

How many different ways are there to select '**me**' ?

```
<div class=great-grandparent>
        <div class=grandparent>
                <div class=parent>
                        <div class=me>
                                <div class=child> </div>
                        </div>
                        <div class=sibling>
                                <div class=niece> </div>
                                <div class=nephew> </div>
                        </div>
                </div>
        </div>
</div>
```

Parents & Childs

How many different ways are there to select '**me**' ?

```
<div class=great-grandparent>
        <div class=grandparent>
                <div class=parent>
                        <div class=me>
                                <div class=child> </div>
                        </div>
                        <div class=sibling>
                                <div class=me> </div>   <!--IMPOSTER-->
                                <div class=nephew> </div>
                        </div>
                </div>
        </div>
</div>
```

- Class Hierarchy
- CSS Variables
- Basic Interactions: A CSS Approach

Declaring Variables

Declare variables in the :root class:

```
:root {
    --var-name: property;
}
```

Reference variables in main css:

```
.cool-variable-class {
    color: var(--var-name);
}
```

Variables Usage

When we declare a variable initially:

```
:root {
    --grey: #7a7a7a;
}
```

We can use it in multiple classes for multiple properties.

```
.cool-variable-class {
    color: var(--grey);
}

.cool-variable-class {
    background-color: var(--grey);
}
```

*we will learn how to reassign variables later with JQuery

- Class Hierarchy
- CSS Variables
- Basic Interactions: A CSS Approach

Basic Interactions

Activated when cursor **hovers** over element.

```
.element:hover
```

Activated when cursor **clicks (and holds)** element.

```
.element:active
```

In both cases, properties will be reset when user un-hovers or releases click.

Interactions By Class

We define the following styles:

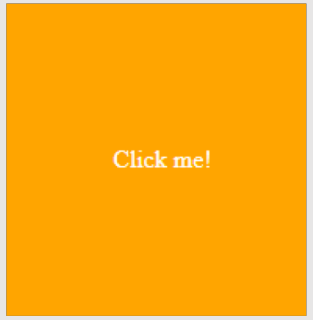
```
.click-me {
    background: orange;
    color: white;
    text-align: center;
    line-height: 200px;
    width: 200px;
    height: 200px; }
```

```
.click-me:active {
    background: blue;
}

.click-me:active::before {
    content: “I am clicked”;
}
```

What is the result when applying it to the following HTML?

```
<div class=click-me> Click me! </div>
```



Interactions By Tag

We define the following styles:

```
a {
    color: black; }
```

```
a:hover {
    color: yellow;
    text-style: underline; }
```

*Notice how we don't use a . or #

What is the result when applying it to the following HTML?

```
<a href='http://www.poptropica.com/' > Some Fun Games </a>
```

Affects every < a > tag

Homework Ideas

- Reorganize HTML structure to encase similar content into the same parenting divs.
- Add **hover** and **active** interactions to elements.

Live Demo