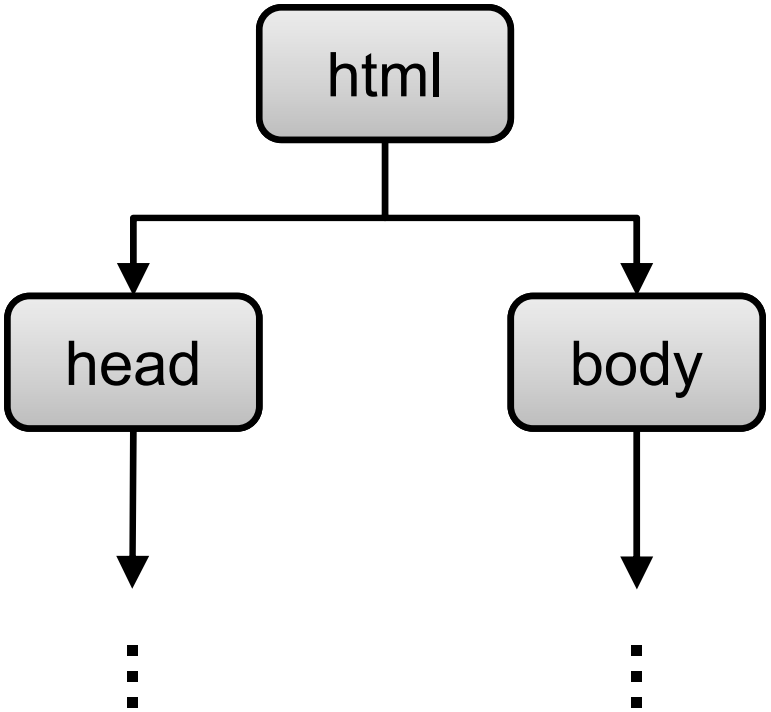
COMP4021
Internet Computing

More on CSS and the DOM

David Rossiter & Gibson Lam

The DOM

- We have already looked at what the DOM (Document Object Model) is
- It's a tree structure which represents the HTML stored inside the browser's memory



Accessing Attributes Using JavaScript

- After you find what you are looking for in the DOM, it's easy to do something with attributes:
  - You can read an attribute by:

```
node.getAttribute("name");
```

  - You can change an attribute by:

```
node.setAttribute("name", "new value");
```

The name of the attribute e.g. color

node points to the HTML node

Example of Changing an Attribute Using JS

- Here the list type of the `<ol>` element is adjusted

```
document.
  getElementsByTagName("ol")[0].
    setAttribute("type", "a");
```

a. Breakfast **$15.00**
b. Lunch **$25.00**
c. Dinner **$50.00**
  - Main course **$30.00**
  - Desert **$20.00**

The list type is changed from 1, 2, 3... to a, b, c…

```
<ol type="a">
  …
</ol>
```

This gives you a, b, c…

Applying the Same CSS Rule to Multiple Things

- Here is an example:

```
ul, ol { color: red; }
```

put a comma between selectors

  – All unordered lists and all ordered lists will have red text

Using Class and Id

```
<p class="fun">
...
</p>
```

- Paragraphs which have the class *fun* will have red text:

```
p.fun { color: red; }
```

```
<p id="fun">
...
</p>
```

- A paragraph which has the id *fun* will have red text:

```
p#fun { color: red; }
```

Starting HTML

- This is the HTML used for the following examples

```
<ol>
  <li>Breakfast <b>$15.00</b></li>
  <li>Lunch <b>$25.00</b></li>
  <li>Dinner <b>$50.00</b>
    <ul>
      <li>Main course
        <b>$30.00</b></li>
      <li>Desert
        <b>$20.00</b></li>
    </ul>
  </li>
</ol>
```

Combinators

- Here we look at advanced style rules called combinators (=combining things)
- There are four types of combinators:
  1. Descendant combinators
  2. Child combinators
  3. Sibling combinators
  4. Adjacent sibling combinators

- *Don't worry about the boring names for these things, you only need to understand what the code does!*

Descendant Combinators

- Here is an example:

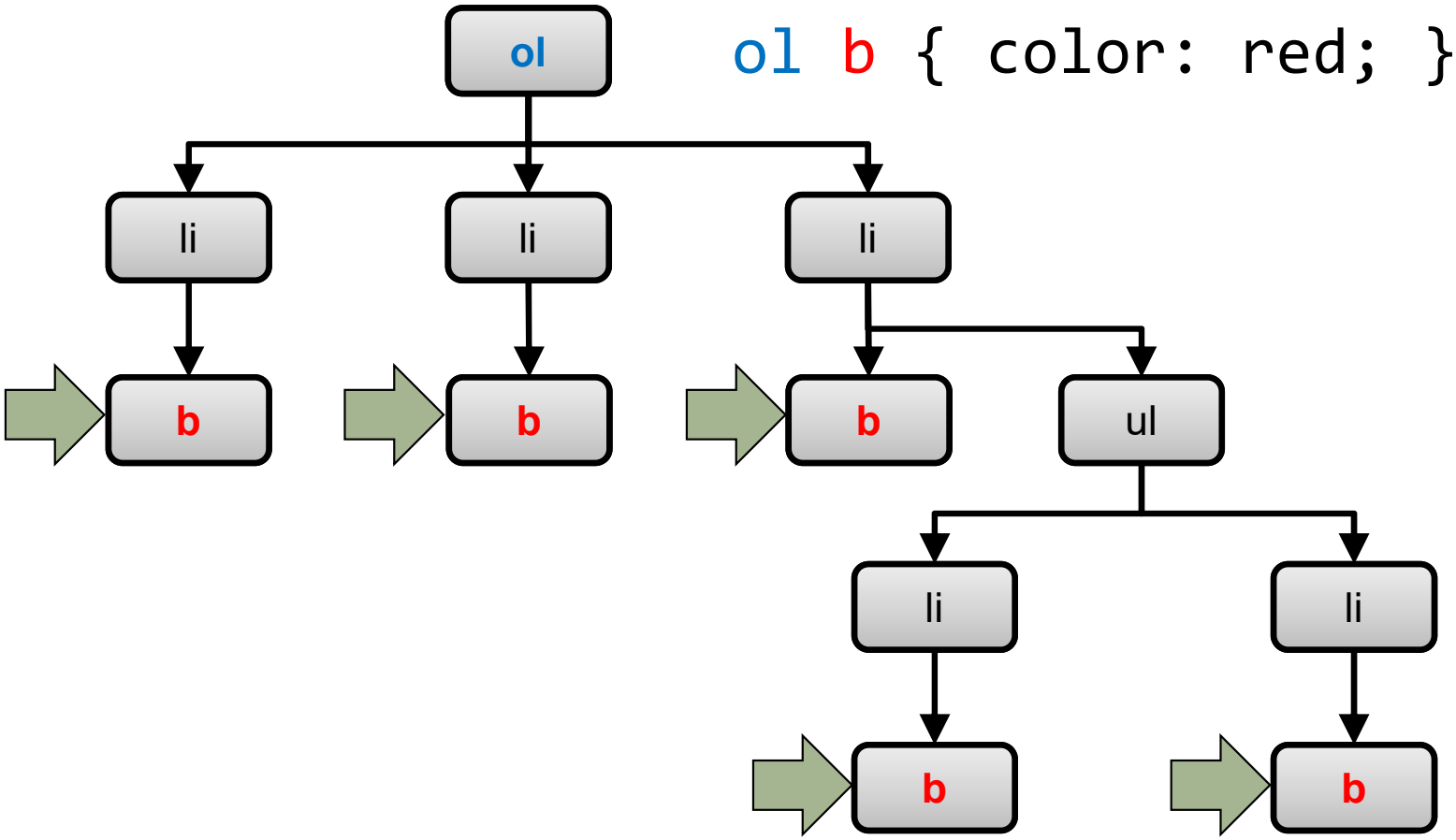
```
ol b { color: red; }
```

  – It changes all <b> elements, which are descendants of an <ol> element, to red

no need to be directly underneath <ol>, anywhere underneath is okay

Descendant Combinator Example

```
ol b { color: red; }
```



Descendant Combinator Example

- This is the result

```
<ol>
  <li>Breakf <b>$15.00</b></li>
  <li>Lu <b>$25.00</b></li>
  <li>Dir <b>$50.00</b>
    <ul>
      <li>Main course
        <b>$30.00</b></li>
      <li>Desert
        <b>$20.00</b></li>
    </ul>
  </li>
</ol>
```

1. Breakfast **$15.00**
2. Lunch **$25.00**
3. Dinner **$50.00**
   - Main course **$30.00**
   - Desert **$20.00**

- *Lines and arrows have been added to emphasise the parts that have changed*

Multiple Levels of Combinators

- Combinators can be put in a series
- For example:

```
div.calendar .day b { color: red; }
```

  – It means that all `<b>` elements,
    that are descendents of any element
      with the class `day` ,
    that are descendents of a
      with the class `calendar` ,
    are changed to red

div & .calender中间没有空格，他们是一个整体，代表：有calender class的div。如果有空格，就是descendent的关系，代表：any element with class of calender underneath any div element

Child Combinators

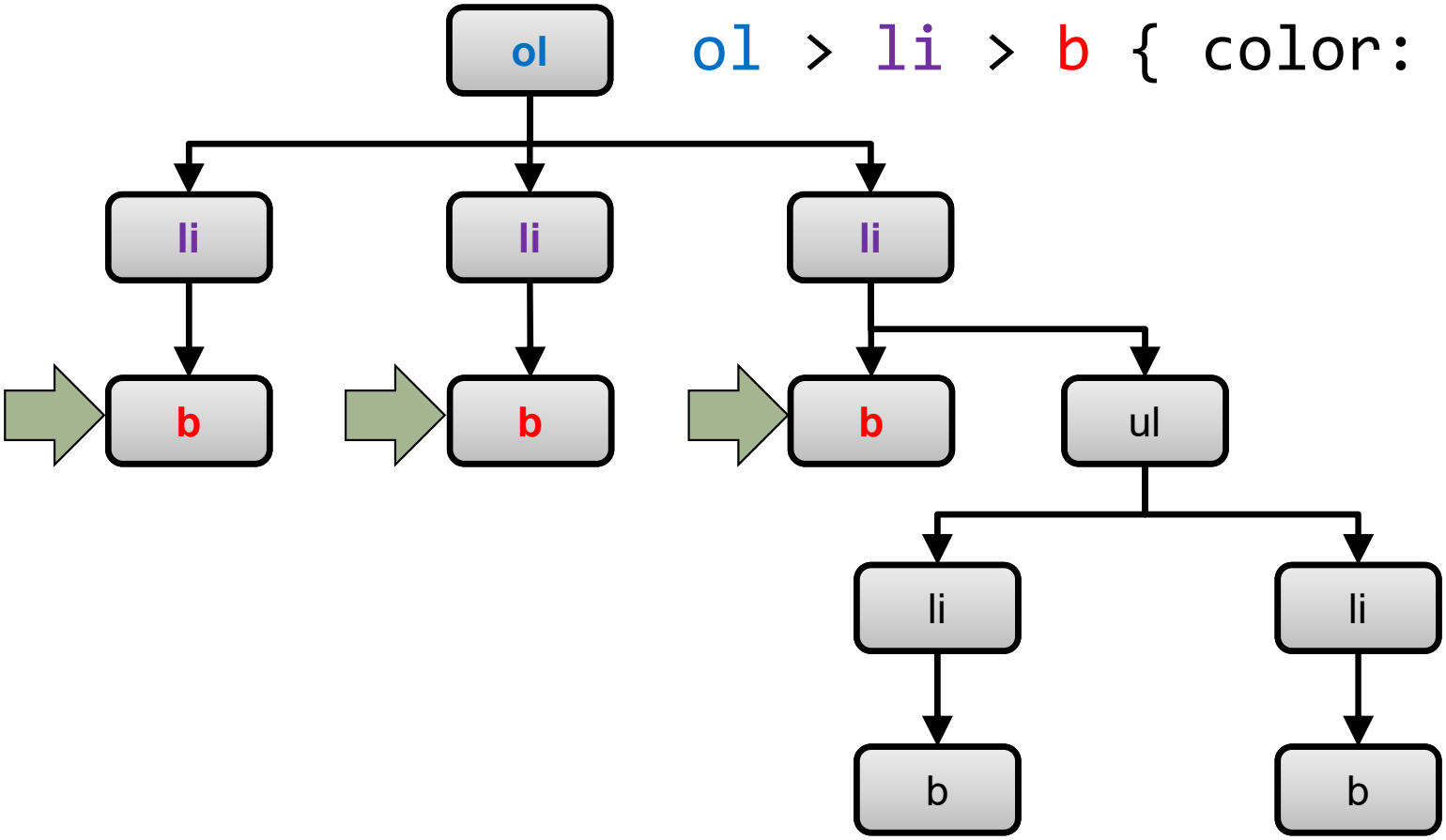
- Here is an example:

```
ol > li > b { color: red; }
```

  – It says that all `<b>` elements, that are children of an `<li>` element, which is a child of an `<ol>` element, are turned to red

Child Combinator Example

```
ol > li > b { color: red; }
```



Child Combinator Example

- This is the result

```
<ol>
  <li>Breakf <b>$15.00</b></li>
  <li>Lu <b>$25.00</b></li>
  <li>Dir <b>$50.00</b>
    <ul>
      <li>Main course
        <b>$30.00</b></li>
      <li>Desert
        <b>$20.00</b></li>
    </ul>
  </li>
</ol>
```

1. Breakfast **$15.00**
2. Lunch **$25.00**
3. Dinner **$50.00**
   - Main course **$30.00**
   - Desert **$20.00**

- *Lines and arrows have been added to emphasise the parts that have changed*

Sibling Combinators

- Here is an example:

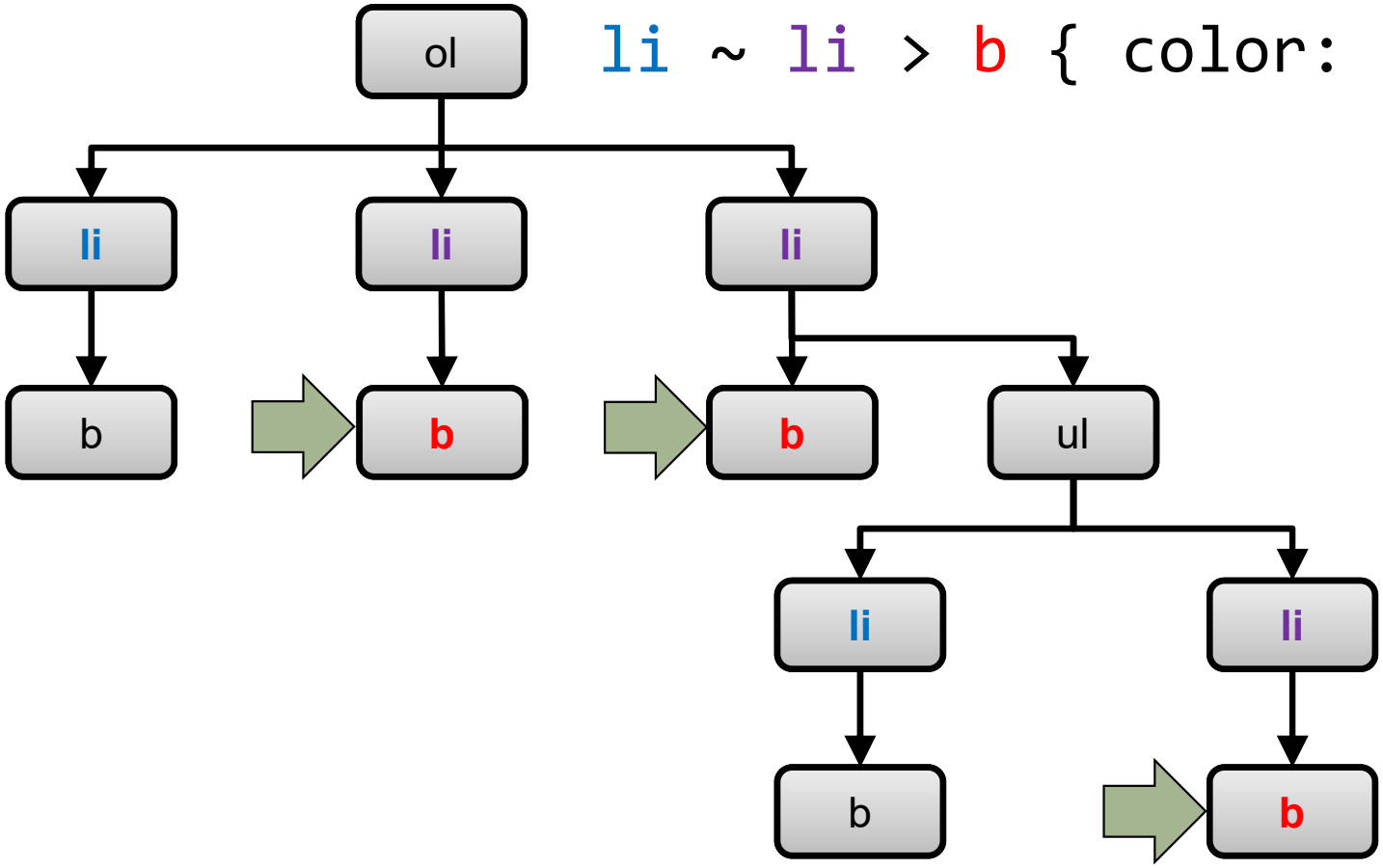
```
li ~ li > b { color: red; }
```

  – This rule changes all `<b>` elements,
    that are children of an `<li>` element,
    which are siblings of a preceding `<li>` element,
    to red

preceding 规定：在DOM structure里面左边有兄弟是<li>，如果左边没有，仅仅右边有兄弟是<li>那不算

Sibling Combinator Example

```
li ~ li > b { color: red; }
```



Sibling Combinator Example

- This is the result

```
<ol>
  <li>Breakfast <b>$15.00</b></li>
  <li>Lu <b>$25.00</b></li>
  <li>Di <b>$50.00</b>
    <ul>
      <li>Main course
        <b>$30.00</b></li>
      <li>Desert
        <b>$20.00</b></li>
    </ul>
  </li>
</ol>
```

1. Breakfast **$15.00**
2. Lunch **$25.00**
3. Dinner **$50.00**
   - Main course **$30.00**
   - Desert **$20.00**

- *Lines and arrows have been added to emphasise the parts that have changed*

Adjacent Sibling Combinators

- Here is an example:

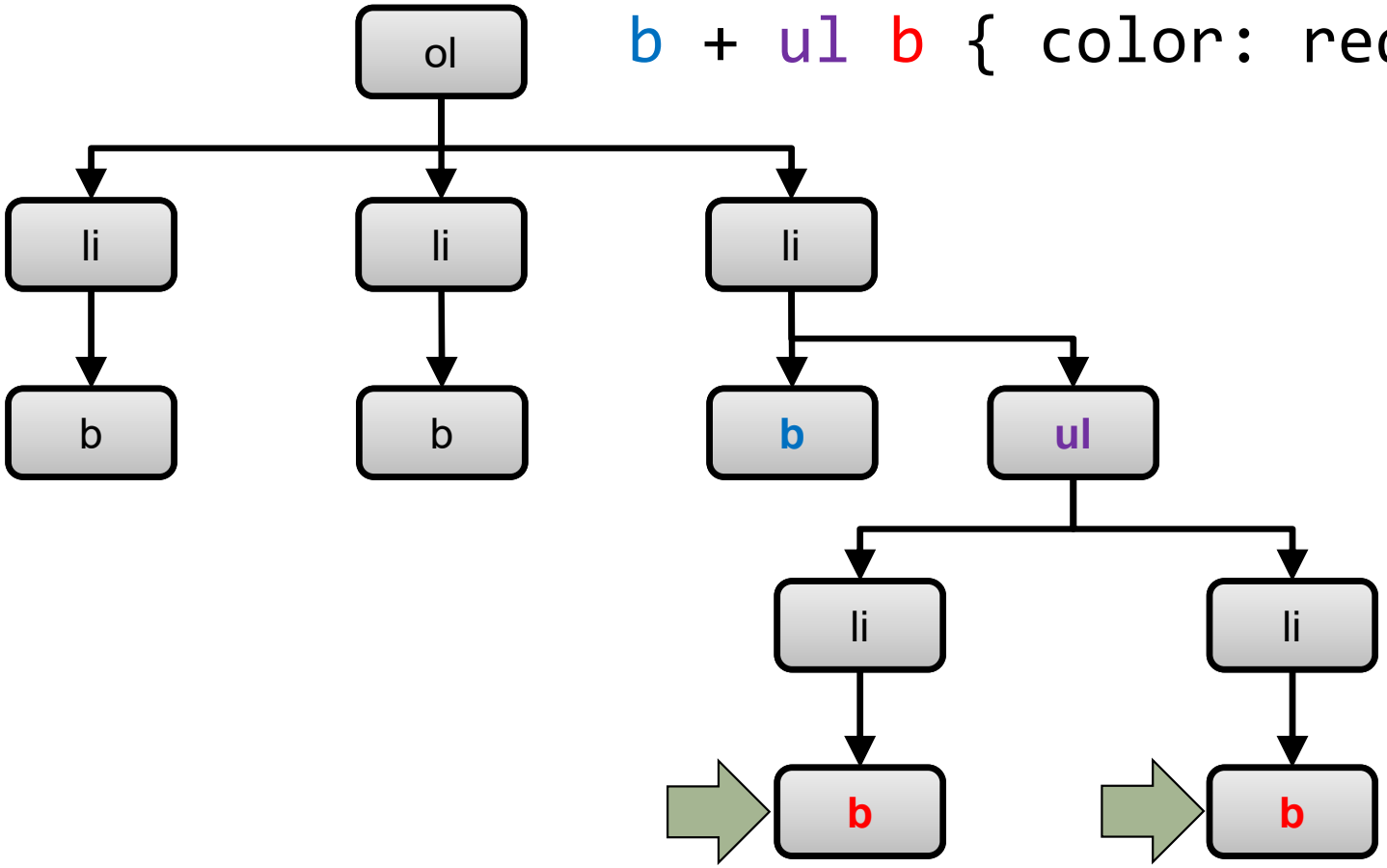
```
b + ul b { color: red; }
```

  - The rule changes the `<b>` elements, that are under an `<ul>` element, which is the **adjacent sibling** of a `<b>` element, to red

  - 'Adjacent' means 'next to'

Adjacent Sibling Example

```
b + ul b { color: red; }
```



Adjacent Sibling Example

- This is the result

```
<ol>
  <li>Breakfast <b>$15.00</b></li>
  <li>Lunch <b>$25.00</b></li>
  <li>Dinner <b>$50.00</b>
    <ul>
      <li>Main course
        <b>$30.00</b></li>
      <li>Desert
        <b>$20.00</b></li>
    </ul>
  </li>
</ol>
```

1. Breakfast **$15.00**
2. Lunch **$25.00**
3. Dinner **$50.00**
   - Main cou **$30.00**
   - Desert **$20.00**

- *Lines and arrows have been added to emphasise the parts that have changed*